

rf

Ein neuer Technikbegriff? Zur Kritik des panalgorithmischen Programms

Rainer Fischbach

[Überarbeitete Fassung vom 30.10.1995]

Adresse des Autors:
Hirtenbrünnle 25 • D-72 181 Starzach
Tel (07478) 2149 • Fax (07478) 2148
Email rf@ba-stuttgart.de

© Rainer Fischbach 1995

Das panalgorithmische Programm

Es scheint ein Kulturge-setz zumindest der neuzeitlichen westlichen Gesellschaften zu sein, daß die Technik, die diese zu einer bestimmten Zeit als die dominierende wahrnehmen, in ihren Augen nicht nur als entscheidender Faktor von Wohlstand und Fortschritt sondern auch als Schlüssel zur Technik überhaupt, ja zur Welt als ganzer erscheint.¹ Die Uhrwerksmetaphorik des 17. und 18. Jahrhunderts illustriert diesen Sachverhalt ebenso wie die Dampfmaschinenrhetorik des 19. Jahrhunderts. Der Panchemismus eines Carl Bosch, der in der Hochdrucksynthese eine Meistertechnik sah, die die Lösung aller Menschheitsprobleme in sich schloße, erscheint demnach als adäquate Technikideologie der IG-Farben-ära der deutschen Geschichte.² Man mag es geradezu als Zeichen für das Einmünden des deutschen Sonderweges, für den nicht zuletzt die IG-Farbenindustrie stand, in den Mainstream der westlichen Geschichte lesen, daß die Ideologiebildung zum Gegenstandsbereich der Technik im deutschen Sprachraum von denselben Phänomenen ausgeht, wie in den anderen westlichen Ländern auch: denen der Informationstechnik und der sogenannten neuen Medien. Davon macht auch die ebenfalls auf – wenn auch weniger destruktive – Sonderwege spezialisierte Schweiz keine Ausnahme.

1. Ich habe mich mit dem Begriff der Schlüsseltechnik bzw. -technologie unter besonderer Berücksichtigung der Informationstechnik bereits an anderem Ort auseinandergesetzt. Vgl. Rainer Fischbach: ›Informationstechnik – Mikroelektronik? Vom Wert und Unwert sogenannter Schlüssel- und Zukunftstechnologien‹. *Johann Beckmann-Journal*, 1995.

2. Vgl. Joachim Radkau: *Technik in Deutschland: Vom 18. Jahrhundert bis zur Gegenwart*. Frankfurt am Main: Suhrkamp, 1989 (Edition Suhrkamp: Neue Folge; Bd. 536), S. 254 ff.

Unter dem Titel *Die Herrschaft der Regel* legt Bettina Heintz³ einen Text vor, der nicht weniger beansprucht, als einen ebenso neuen wie radikalen Technikbegriff zu formulieren. Daß dieses Unternehmen sich im Untertitel als Beitrag »zur Grundlagengeschichte des Computers« ausgibt, hat damit zu tun, daß die Autorin in der Computertechnik bzw. in dem, was sie dafür hält, den Zugang zum bisher verborgenen Wesen der Technik finden zu können glaubt. Die Essenz nicht nur jener sondern jeglicher Technik sieht sie im Algorithmus, wobei dessen Essentialität für sie in der Tatsache liegt, daß derselbe Algorithmus durch verschiedene Medien implementierbar ist, diese selbst also völlig unwesentlich seien:

»Wenn es aber theoretisch gesehen nicht von Bedeutung ist, auf welche Weise ein Algorithmus konkret realisiert ist, ob er von einem Menschen ausgeführt wird oder von einer Maschine, dann geht damit auch eine Bedeutungsverschiebung im Begriff des »Technischen« einher. Das Entscheidende am Technischen ist dann nicht die Maschinerie, sondern das Algorithmische. »Maschinenmäßig« ist in dieser Sicht alles, was nach einer klaren Vorschrift abläuft – unabhängig davon, wie es konkret implementiert ist.«⁴

An dieser Formulierung fällt eine gewisse Unschärfe auf: ist gemeint, daß »das Technische« als vages Abstraktum »im Algorithmischen« gründe, oder, was eine wesentlich präzisere und konkret falsifizierbare Aussage wäre, daß es jeweils ein bestimmter Algorithmus sei, der das Spezifikum einer Maschine ausmache? Im Zusammenhang ihrer Argumentation wird deutlich, daß Bettina Heintz letzteres sagen möchte, obwohl sie das nirgendwo explizit tut. Im anderen Falle könnte es durchaus so sein, daß – was eine wesentlich schwächere Aussage darstellte – Maschinenhaftigkeit, Technizität – weshalb auch immer – in jedem Falle mit Algorithmischem einhergingen, ohne daß dessen Ausprägung die Differentia spezifika einer Maschine bzw. eines Maschinentyps ausmache.

Im Computer kulminierte eine Entwicklung, die, vom Anfang dieses Jahrhunderts an, alle gesellschaftlichen Sphären, die theoretische nicht minder als die materielle Produktion zunehmend durchdringend, zur *Herrschaft der Regel*, also zur Dominanz rigider Mechanismen und formaler Prinzipien geführt hätte. Dabei könne die formalistische Mathematik, die im Deutschland des ersten Drittels dieses Jahrhunderts ihren Ausgang vom axiomatischen Programm David Hilberts nahm, besondere Aufmerksamkeit nicht nur deshalb beanspruchen, weil in ihr der Ursprung des Computers liege, sondern auch, weil dort modellhaft in kristalliner, durch keinen Weltbezug kontaminierter Form sich jene Prinzipien der formalen Rationalität, die gleichzeitig ihren Durchbruch in der soziokulturellen Realität erlebt hätten, faßbar geworden wären. Sie formuliere den Generalnenner der Moderne:

3. Bettina Heintz: *Die Herrschaft der Regel: Zur Grundlagengeschichte des Computers*. Frankfurt am Main: Campus, 1993.

4. A. a. O., S. 258.

»Im Hilbertschen Formalismus haben sich die Zeichen aus der Welt zurückgezogen. Sie sind nur noch Partikel eines in sich geschlossenen semiotischen Systems. Die Moderne geht einher mit dem Verzicht auf Repräsentation. Dies gilt gleichermaßen für die moderne Mathematik wie für die Moderne in Kunst, Musik und Literatur. Die Hilbertsche Mathematik hat den Bruch mit dem Repräsentationsmodell endgültig vollzogen. Die Unterscheidung zwischen Zeichen und Bezeichnetem, die bis dahin unangetastet blieb, löst sich in der mathematischen Moderne auf. Was bleibt, sind die Zeichen, die, wenn überhaupt, bloß auf andere Zeichen verweisen, und Regeln, die bestimmen, wie mit ihnen umzugehen ist.

Die Mathematik der Moderne ist die symbolische Welt, in der der Computer gedacht wurde. 1936 hat der englische Mathematiker Alan M. Turing in einer berühmten These die Behauptung aufgestellt, daß jedes Handeln, das einer klaren Vorschrift folgt, auch von einer Maschine ausgeführt, d.h. mechanisiert werden kann. [...] 1936, als Turing seine Arbeit schrieb, gab es noch keine Computer, und Turing hat auch nicht an richtige Maschinen gedacht, als er seine Papiermaschine konzipierte. Dennoch hat er mit seinem Maschinenmodell das Grunddesign von Digitalcomputern vorweggenommen. Alle Digitalcomputer, die seither entwickelt wurden, sind gleichermaßen gerätetechnische Realisierungen von Turings »symbolischer« Maschine.⁵

Worauf die Autorin zielt, ist eine »Soziologie der Computerisierung«,⁶ die die formalistische Mathematik ebenso wie die nach ihrer Meinung aus ihr hervorgegangene Computertechnik als Kristalisationsformen einer universalen Tendenz zu Entinhaltlichung, Formalisierung und Rationalisierung, die als geheimes Wesen in aller Technik wirke und der Moderne ihren Stempel aufdrücke, ausweist. »Läßt sich ein Zusammenhang ausmachen zwischen der Moderne in der Mathematik und der Modernisierung im soziokulturellen Bereich?«, fragt sie, um ihre Frage sogleich zu bejahen, indem sie zu zeigen versucht,

»wie sehr das mathematische Denken von David Hilbert und Alan Turing von außermathematischen Erfahrungen geprägt worden ist – von jenen Erfahrungen, die man in der Soziologie zu dieser Zeit unter dem Begriff »Rationalisierung« diskutierte.«⁷

An anderem Ort⁸ versucht sie, ihren Anspruch um eine kunsttheoretische Facette zu erweitern. Dort erkennt sie in den Revuegirls der 20er Jahre

»die ästhetische Verkörperung des Rationalisierungsprinzips. Sie waren der körperliche Ausdruck einer Gesellschaft, die nach rationalen Prinzipien funktioniert und in der das Individuum auf ein Masseteilchen reduziert ist. [...] Dem Rythmus der Musik entspricht der Takt des Fließbandes; den Beinen der Girls entsprechen die Hände in der Fabrik; den Instruktionen des Choreographen die Anweisungen des Funktionsmeisters. Auf der Bühne wie in der Fabrik werden Bewegungen gemessen, analysiert, in ihre Bestandteile zerlegt und nach rationalen oder ästhetischen Prinzipien neu zusammengefügt.«⁹

5. A. a. O., S. 9 f.

6. A. a. O., S. 10.

7. A. a. O., S. 11.

8. Bettina Heintz: »Das Fließband im Kopf: Über Zahlenmanufakturen, Fabrikarbeit und Tanzrevueen«. *Wechselwirkung*, April 1995, S. 50–55.

9. A. a. O., S. 54.

In der Komposition von zuvor analytisch isolierten Bestandteilen nach Gesichtspunkten von Rationalität und Effizienz sieht Heintz das Wirken des algorithmischen Prinzips; was sie dazu verleitet, die Begriffe *Maschine*, *Verfahren* und *Algorithmus* zu identifizieren.¹⁰ Diesen algorithmischen Verfahrens- und Maschinenbegriff glaubt sie auf das Werk Alan Turings zurückführen zu können und beansprucht,

›daß sich auf der Basis von Turings algorithmischem Maschinenbegriff präziser formulieren läßt, was mit »mechanischem« Verhalten genau gemeint ist. Mit seiner konzeptionellen Entflechtung von Algorithmus und gerätetechnischer Apparatur hat Turing eine theoretische Grundlage bereitgestellt für die Entwicklung eines Maschinenbegriffs, der das Maschinenhafte nicht mehr über dessen materielle Beschaffenheit definiert, sondern über das Verhalten. Aus dieser Perspektive betrachtet beschränkt sich der Gegenstandsbereich der Techniksoziologie nicht auf technische Artefakte, sondern umfaßt alle Prozesse, die sich algorithmisch beschreiben lassen, ungeachtet wie und durch wen sie realisiert sind.«¹¹

Daß das Implementationsmedium eines Algorithmus gleichgültig ist, bedeute, daß ›die Grenzlinie zwischen Mensch und Maschine brüchig‹¹² werde; womit Heintz sich in eine Traditionslinie des Theoretisierens über Technik und Computer stellt, deren Ziel die Auslöschung jeglicher Spuren eines emphatischen Begriffs von Vernunft ist, der darauf besteht, daß diese sich in ihren formalen, instrumentellen Aspekten nicht erschöpfe. Sie knüpft dabei explizit an Bammé u. a. an,¹³ die bereits 1983 die These vertreten, ›es sei gerade die Maschine, die neue Maschine, die den Menschen vom Tier unterscheidet.«¹⁴

Mit der Rede von ›der neuen Maschine‹ bedienen Bammé und seine Koautoren sich jener begriffsimperialistischen Metaphorik, der nach der Devise ›der Algorithmus ist die Maschine‹,¹⁵ die immerhin den Vorzug hat, anders als die vage Rede vom ›Technischen‹ und von ›Algorithmischen‹, deren Heintz sich befleißigt, klar zu sagen, was sie meint, alles und jedes, zuletzt auch der menschliche Intellekt, als Maschine gilt. Was Bettina Heintz dagegen versucht, ist, dieser Metaphorik ein wissenschaftssoziologisches Fundament zu bauen. Sie ist dabei weniger explizit in ihren Empfehlungen als Bammé und Kollegen, die folgendes vorschlagen:

›Die Annahme, daß abstraktes, rationales Denken das eigentlich Menschliche sei, führt letztlich zur Konsequenz, daß es das Maschinenhafte sei, was den Menschen ausmache, sonst müßten die Spezifika des Menschen auf ganz anderen Gebieten gesucht werden. [...] Die Abspaltung von maschinisierbaren Denkfunktionen beinhaltet durchaus befreiende Momente. Sie zwingt uns, unsere Wertmaß-

10. Vgl. a. a. O., S. 55.

11. Heintz, *Die Herrschaft der Regel*, a. a. O., S. 12.

12. A. a. O.

13. Vgl. A. a. O., S. 258.

14. Arno Bammé u. a.: *Maschinen-Menschen, Mensch-Maschinen: Grundrisse einer sozialen Beziehung*. Reinbek: Rowohlt, 1983 (rororo; Bd. 7698), S. 18.

15. A. a. O., S. 145.

stäbe zu verändern: präzises, eindeutiges, abstraktes Denken können wir getrost der Maschine überlassen. [...] Der Mensch verliert dadurch substantiell nichts von seiner Persönlichkeit; er entlastet sich von historisch überflüssigem Ballast. Dieser Aspekt wird von der konservativen Kritik ebenso wie von denen, die eine diffuse Angst vor den Fähigkeiten der Computer empfinden, immer übersehen.¹⁶

Das ist natürlich eine besonders ausgebuffte Reklame für die Marke *Zeitgeist Lite*: Ihr Genuß stellt sich als Konsequenz einer tiefen Einsicht dar; wonach, wer die Zeichen der Zeit erkannt habe und den Möglichkeiten der modernen Technik furchtlos, entspannt und aufgeschlossen gegenüberstehe, nichts daran finde, auch mal auf ätzendes, präzises Denken zugunsten jenes angenehm benebelnden Gesöffs zu verzichten. Die folgenden Abschnitte tun dar, daß Heintz, Bammé u. a. doch zu früh einen zu kräftigen Schluck aus der Pulle genommen haben.

Damit sind die Ingredienzien dessen, was ich das *panolgorithmische Programm* nenne, versammelt. Die folgende Liste gibt eine grobe Skizze. Das Besondere an dem Beitrag von Bettina Heintz ist, daß sie das Grundsatzzprogramm (Thesen 1-3) um unterstützende wissenschaftssoziologische Aussagen (Thesen 4-7) ergänzt:

1. Jedes technische Artefakt, jedes Verfahren sei wesentlich durch einen Algorithmus konstituiert: dieser Algorithmus *sei* das eigentliche Artefakt, das eigentliche Verfahren.
2. Im Universalrechner, der *universellen Maschine*, deren Verhalten nicht mehr durch einen in der Konstruktion geronnenen Algorithmus festgelegt, sondern durch beliebige Algorithmen frei bestimmbar sei, offenbare sich das algorithmische Wesen der Technik.
3. Der derart entsubstantialisierte Technikbegriff läßt kaum etwas übrig, was *keine* Technik mehr wäre: Alles, was sich präzise beschreiben lasse, sei algorithmisierbar und damit auch technisch.
4. In Gestalt des Formalismus, der sich in den 20er Jahren siegreich gegen den Intuitionismus durchgesetzt habe und zur ›Arbeitsphilosophie des »working mathematician« geworden sei, habe sich das algorithmische Prinzip der Mathematik, in der seither der Geist der Maschine genauso herrsche wie in der Fabrik, bemächtigt und in ihr modellhaft auskristallisiert.
5. Die innermathematische Auseinandersetzung zwischen Formalisten und Intuitionisten spiegele die soziale und politische Auseinandersetzung zwischen Moderne und Gegenmoderne in den 20er Jahren.
6. Der Siegeszug des mathematischen Formalismus entspreche einer umfassenden Tendenz hin zu Entinhaltlichung, Formalisierung und Rationalisierung, die sich in allen Bereichen, von der soziologischen Theoriebildung über die Fabrikarbeit bis hin zum Tanz manifestiere.

16. A. a. O., S. 158.

7. Im Computer feiere das algorithmische Prinzip, das in der Maschinenwelt bisher nur in materiell verunreinigter Form vorgekommen sei, geläutert durch das Purgatorium der formalistischen Mathematik und aus dieser quasi neugeboren und zu sich selbst gekommen, seine Auferstehung in ebenso reiner wie konkreter Form.

Dieses Programm basiert auf Mißverständnissen, die teils einer weitgehenden Unkenntnis der betreffenden Gebiete, teils einer unscharfen, allenfalls metaphorischen Begriffsbildung geschuldet sind:

1. Die Gleichsetzung von Algorithmus und Maschine wird weder dem Begriff der Maschine noch dem des Algorithmus gerecht.
 2. Der Computer ist *keine* universelle Maschine.
 3. Die Universalität des panalgorithmischen Programms beruht auf der falschen Identifikation von Zeichen, Zeichenträger und Bezeichnetem.
 4. Der Formalismus definiert kein Modell mathematischer Arbeit: Er hat weder die Rolle noch die Bedeutung, die Bettina Heintz ihm beimißt.
 5. Der Kampf zwischen Formalisten und Intuitionisten ist als Reflex eines Kampfes zwischen Moderne und Gegenmoderne nicht zu verstehen.
 6. Die Tendenz zur Rationalisierung war in den 20er Jahren nicht so universell, wie Heintz behauptet. Ihre Beispiele sind nicht evident.
 7. Der Computer ist keine Emanation des mathematischen Formalismus.
- Die Grundlagengeschichte des Computers, die Bettina Heintz ausbreitet, ist eine wissenschaftshistorische Fälschung. Der Imperialismus, d. h. die angemaßte Universalität des panalgorithmischen Programms steht sehr wohl in der Tradition eines Carl Bosch und seines Panchemismus.

Formalismus und mathematische Grundagentheorie

Wichtigster Gewährsmann in Bettina Heintz fiktiver Geschichte der modernen Mathematik ist David Hilbert. Hilbert habe die antiintuitive Wende in der Mathematik vollzogen, indem er diese auf ein Operieren mit formalen Objekten ohne Bedeutung reduziert habe.

Hilbert trat tatsächlich mit einem beweistheoretischen Programm hervor, dem stark formalistische Züge anhaften. Gleichwohl gibt es gute Gründe dafür, anzunehmen, daß Hilberts Verständnis von Mathematik nicht so schlicht war, daß er glaubte, diese reduziere sich auf ein Spiel mit leeren Formeln. Dagegen spricht schon seine umfassende mathematische Praxis. Sein Formalismus ist tatsächlich nicht strategischer sondern taktischer Natur und soll dazu dienen, den alles andere als formalistischen Gehalt der Mathematik, dem er hohe Bedeutung beimaß, zu retten. Daß Hilbert kein Schmalspurformalist war, der der Anschauung keinen Platz in der Mathematik zugebilligt hätte, bezeugt schon die Tatsache, daß er Hauptautor eines Werks mit dem Titel *Anschauliche Geometrie*¹⁷ ist. Im

17. David Hilbert, Stefan Cohn-Vossen: *Anschauliche Geometrie*. Berlin: Springer, 1932

Vorwort dieses Werkes artikuliert er ein klares Bewußtsein von der Bedeutung der Anschauung:

»In der Mathematik wie in aller wissenschaftlichen Forschung treffen wir zweierlei Tendenzen an: die Tendenz zur Abstraktion – sie sucht die *logischen* Gesichtspunkte aus dem vielfältigen Material herauszuarbeiten und dieses in systematischen Zusammenhang zu bringen – und die andere Tendenz, die der Anschaulichkeit, die vielmehr auf ein lebendiges Erfassen der Gegenstände und ihre *inhaltlichen* Beziehungen ausgeht.

Was insbesondere die Geometrie betrifft, so hat bei ihr die abstrakte Tendenz zu den großartigen systematischen Lehrgebäuden der algebraischen Geometrie, der RIEMANNschen Geometrie und der Topologie geführt, in denen die Methoden der begrifflichen Überlegung, der Symbolik und des Kalküls in ausgiebigem Maße zur Verwendung gelangen. Dennoch kommt auch heute dem anschaulichen Erfassen in der Geometrie eine hervorragende Rolle zu, und zwar nicht nur als einer überlegenen Kraft des Forschens, sondern auch für die Auffassung und Würdigung der Forschungsergebnisse.¹⁸

Sollte Hilbert etwa bereit gewesen sein, auf »eine überlegene Kraft des Forschens« zu verzichten? Bestimmt nicht. Dazu war ihm die Mathematik zu wichtig. Was eine präzise Auseinandersetzung mit dem von Heintz konstruierten gedanklichen Gebilde, das sie mathematischen Formalismus nennt, erschwert, ist der Umstand, daß sie nicht genau sagt, welchen Realitätsgehalt sie für dieses beansprucht. Vielmehr gibt es dazu widersprüchliche Aussagen. So tastet sie sich zunächst sehr vorsichtig an ihre eigenen Thesen heran, so, als ob sie ihnen mißtraute:

»Wenn ich im folgenden von »Formalismus« spreche, beziehe ich mich auf eine spezifische *Auffassung* der Mathematik, so wie sie in grundlagentheoretischen Arbeiten formuliert wurde, *nicht* auf die alltägliche Praxis der Mathematiker und Mathematikerinnen. Der »working mathematician« betreibt Mathematik nicht so, wie der Formalismus die Mathematik beschreibt. So spielen z. B. Erfahrungswissen und Intuition im Alltag der Mathematik eine beträchtliche Rolle.«¹⁹

Doch mit der Vorsicht ist es bald vorbei; was sie schon wenige Seiten weiter zur Behauptung des Gegenteils verführt:

»Der Formalismus hatte sich durchgesetzt, als Arbeitsphilosophie des »working mathematician« gewissermaßen, unbeschadet seiner nicht eingelösten Versprechen. Als Alan Turing Mitte der 30er Jahre über seinen Algorithmus-Begriff nachzudenken begann, war die formalistische Auffassung der Mathematik bereits fest etabliert.«²⁰

Und später noch einmal die starke These, daß der Formalismus zu einer die Praxis der Mathematiker bestimmenden Doktrin geworden sei:

»Als Hilbert in Bologna seinen Vortrag hielt [1928, R.F.], hatte sich die formalistische Auffassung der Mathematik als handlungsleitendes Modell zu einem großen Teil bereits durchgesetzt.«²¹

(Grundlehren der mathematischen Wissenschaften; Bd. xxxvii).

18. A. a. O., S. v.

19. Heintz, a. a. O., S. 16.

20. A. a. O., S. 62.

21. A. a. O., S. 200.

Die Autorin befindet sich hier in einem Dilemma: Wenn sie eingestehen würde, welche beschränkte Bedeutung dem Formalismus in der mathematischen Praxis zukommt, wie fiktiv vor allem das von ihr aus isolierten, oft aus zweiter Hand genommenen Literaturzitaten konstruierte Gebilde, das sie Formalismus nennt, ist, dann bräche ihr wissenschaftssoziologischer Argumentationsgang schon im Ansatz ein. Deshalb erscheint es ihr zu riskant, ihre Fiktion gleich zu Beginn einer möglichen Konfrontation mit den Fakten zu exponieren; deshalb die anfangs vorsichtige Äußerung zur Bedeutung des Formalismus.

Doch dabei kann es nicht bleiben: Um ihre These von der Emanation des Computers aus der formalistisch mutierten Mathematik und weitergehend, vom mechanischen Charakter aller Rationalität zu stützen, muß Heintz eine entsprechende, explizite Behauptung über die Natur der mathematischen Tätigkeit aufstellen, die das Bindeglied zwischen Mathematik, Computer und dem maschinenhaften Charakter aller Rationalität dingfest macht:

»Bereits Hilbert hat die ehemals idealen Objekte der Mathematik auf profane Zeichen reduziert und der Tätigkeit des Mathematikers den trivialen Sinn von Arbeit gegeben. Aus der Perspektive des Formalismus gleicht das Betreiben von Mathematik einem mechanischen Operieren an und mit diesen Zeichen. Nicht irgendwelche transzendenten Objekte und auch nicht die konstruktiven Schöpfungen des menschlichen Geistes bilden die Gegenstände der Mathematik, sondern physikalisch realisierte Zeichen, banale Tintenstriche auf Papier. [...] Turing hat dem mathematischen Formalismus endgültig eine »industrielle« Wendung gegeben: Hilberts Tintenstriche werden zu Markierungen auf einem Maschinenband, und aus Hilberts mechanischem »Handeln nach Regeln« macht Turing eine maschinelle Operation.«²²

Doch selbst hier scheint sie sich ihrer Sache nicht ganz sicher zu sein: Was heißt es, einer Tätigkeit den »Sinn von Arbeit« – was immer der sei – zu geben, oder was soll damit gesagt sein, wenn sie behauptet, daß Mathematik betreiben dem Operieren mit Zeichen »gleiche«? Alles nur Gleichnis, Metapher, vage Sinngebung, tiefes Geheimnis?

Wie steht es nun mit der tatsächlichen Bedeutung des Formalismus und in welchem Verhältnis steht sie zu dem Bild, das Bettina Heintz von ihm malt? Die Heintzsche Fiktion einer formalistischen Arbeitsphilosophie, die, die mathematische Tätigkeit zum Operieren mit Tintenstrichen degradierend und damit die Erfindung des Computers vorwegnehmend, den maschinenhaften Charakter aller intellektuellen Aktivität offenbart hätte, beruht auf einem dreifachen Irrtum: nämlich daß

1. die formalen Kalküle der modernen Mathematik die Arbeitsweise der modernen Mathematiker reflektierten,
2. daß Zeichen physische Entitäten wären,
3. daß es so etwa gäbe wie bedeutungslose Zeichen.

22. A. a. O., S. 102.

Die Formalisierung mathematischer Theorien, die nicht nur Formalisten sondern auch Platoniker und Intuitionisten betreiben – strittig ist zwischen den Schulen vor allem, *welche* Axiome mit *welchen* Ableitungsregeln unter *welchen* Rechtfertigungen zu formalisieren sind und *welchen* ontologischen Status man dem Ergebnis zubilligt – dient *nicht* dazu, die mathematische Tätigkeit in ein Operieren mit Objekten ohne Inhalt zu verwandeln. Der Kalkül ist *nicht* die wesentliche Arbeitsweise des forschenden Mathematikers – und zwar weder vor noch nach Hilbert. Vielmehr bilden die Kalküle Gegenstände der Metamathematik, der universellen Algebra, der Kategorientheorie, etc. genau so, wie die Zahlen den Gegenstand der Arithmetik und die Figuren den Gegenstand der Geometrie bilden – nicht jedoch ein Modell der mathematischen Arbeit:

›The »logical treatment« of mathematical theories [...] is really of a very different nature. It is actually very difficult to give strict logical proofs of even simple mathematical theorems from the axioms of the mathematical theory concerned. Mathematicians use many definitional abbreviations and long jumps in reasoning, and a clear, convincing mathematical argument is usually no more than a very sketchy recipe for finding the corresponding strict, logical argument in the formal system of proof. The importance of logic for mathematics does not lie in doggedly translating axioms and theorems into formal notation and then supplying the missing minutiae of reasoning. It lies, rather, in answering certain general foundational questions about any mathematical theory.²³

Tatsächlich empfinden Mathematiker den *Archipel Bourbaki* als nicht nur wenig angenehmen sondern auch höchst unfruchtbaren Aufenthaltsort. Kein wirklicher Mathematiker möchte dort wohnen und auch Hilbert tut das nicht. Hilberts Umgang mit dem Formalismus ist taktischer Natur. Der größte Teil seines Werkes gilt der inhaltlichen Mathematik. Die Bedeutung seiner beweistheoretischen Studien besteht eben *nicht* darin, die Arbeit der Mathematiker zu mechanisieren:

›Hilbert was certainly not a fanatic of crude formalism: thus while his problems concerned syntactic properties of formal systems, the solutions were to be given by intuitively correct reasoning, and he explicitly considered any formalization of *this* reasoning unnecessary.²⁴

Diese Einschätzung aus der Feder von Georg Kreisel, eines bedeutenden Grundlagentheoretikers und Kenners des Hilbertschen Werkes, wird unterstützt durch die Darlegung des Themas in der historisch frühesten umfassenden Darstellung der Metamathematik, die Steven Cole Kleene – ebenfalls einer der Großen der mathematischen Grundlagentheorie, auf den wesentliche Beiträge zur Logik und zur Theorie der Berechenbarkeit zurückgehen – vor mehr als vier Jahrzehnten verfaßte:

23. Neil W. Tennant: *Natural Logic*. Edinburgh: Edinburgh University Press, 1990, S. 7.

24. Georg Kreisel: ›Hilberts' Programme«. Paul Benacerraf, Hilary Putnam (Hrsg.): *Philosophy of Mathematics: Selected Readings*, 2. Aufl., Cambridge: Cambridge University Press, 1983, S. 208.

›To Hilbert is due now, first, the emphasis that strict formalization of a theory involves the total abstraction from the meaning, the result being called a *formal system* or *formalism* (or sometimes a *formal theory* or *formal mathematics*); and second his method of making the formal system as a whole the object of a mathematical study called *metamathematics* or *proof theory*.

Metamathematics includes the description or definition of formal systems as well as the investigation of properties of formal systems. In dealing with a particular formal system, we call the system the *object theory*, and the metamathematics relating to it its *metatheory*. From the standpoint of the metatheory, the object theory is not properly a theory at all as we formerly understood the term, but a system of meaningless objects like the positions in a game of chess, subject to mechanical manipulations like the moves in chess. The object theory is described and studied as a system of symbols and of objects built up out of symbols. The symbols are regarded simply as various kinds of recognizable objects. To fix our ideas we may think of them concretely as marks on paper; or more accurately as abstracted from our experience with symbols as marks on paper. (Proof theory must be to some extent abstract, since it supposes arbitrarily long sequences of symbols to be constructible, although the quantity of paper and ink in the world is finite.) The other objects of the system are analyzed only with regard to the manner of their composition out of the symbols. By definition, this is all that a formal system shall be as an object of study for metamathematics.²⁵

Die Metamathematik untersucht die Struktur formaler Gebilde, indem sie von deren Bedeutung *abstrahiert* und nur die *explizit formalisierten* Beziehungen ihrer Elemente betrachtet. Dabei geht sie *intuitiv* vor:

›The metatheory belongs to intuitive and informal mathematics (unless the metatheory is itself formalized from a metametatheory, which here we leave out of account). The metatheory will be expressed in ordinary language, with mathematical symbols, such as metamathematical variables, introduced according to need. The assertions of the metatheory must be understood. The deductions must carry conviction. They proceed by intuitive inferences, and not, as the deductions in the formal theory, by applications of stated rules. Rules have been stated to formalize the object theory, but now we must understand without rules how those rules work. An intuitive mathematics is necessary even to define the formal mathematics. (We shall understand this to mean that the ultimate appeal to justify a metamathematical inference must be to the meaning and evidence rather than to any set of conventional rules. [...])²⁶

Die formalen Systeme reflektieren nicht eine neue Weise Mathematik zu betreiben, sondern bilden den Gegenstand einiger Zweige der intuitiven Mathematik. Die Bausteine der formalen Systeme sind selbst schon durch die Tätigkeit des Intellekts vermittelt. Wie Kleene bemerkt, ist die Vorstellung, daß es sich bei den formalen Objekten um Tintenstriche, Marken auf dem Papier handle, irreführend. Vielmehr sind dies *ideale Objekte* der Art, die Frau Heintz bereits der Obsoleszenz anheimgegeben wähnt. Auch das ›Maschinenband‹ Turings, das sie so sehr fasziniert, ist kein reales Maschinenband sondern – was schon daran erkennbar ist, daß es

25. Steven Cole Kleene: *Introduction to Metamathematics*. Amsterdam: North-Holland, 1952 (Bibliotheca Mathematica; Bd. i), S. 61 f.

26. A. a. O., S. 62.

auf einer Seite unbegrenzt ist – ein solches ideales mathematisches Objekt. Metamathematische Überlegungen wie z.B. der Gödelsche Unvollständigkeitsbeweis handeln von unendlichen Klassen solcher idealer Objekte und unter diesen befinden sich auch welche, die in irgend einer Form – durch Aufschreiben, Codieren auf einem Datenträger, etc. – physisch zu realisieren die im uns bekannten Universum vorhandene Materie nicht ausreicht. Die für den Gödelschen Unvollständigkeitsbeweis zentrale Formel der formalen Arithmetik, deren Interpretation innerhalb der intuitiven Metamathematik die Unbeweisbarkeit eben jener Formel innerhalb dieser Theorie behauptet, hat als Ausdruck des formalen Systems wahrscheinlich noch niemand hingeschrieben und auch die Dezimaldarstellung ihrer Gödelnummer steht uns nicht zur Verfügung. Gödel beweist nur die Konstruierbarkeit dieser Formel in der Theorie.

Der Kern der Leistung Gödels besteht darin, den für die meisten – mit Ausnahmen wie Poincaré und Weyl – seiner Zeitgenossen nicht offensichtlichen, idealen Charakter der Objekte von formalen Systemen zu sehen.²⁷ Genau diese Einsicht ermöglicht ihm die Identifikation der formalen Objekte mit Zahlen und der Beweistheorie mit der Arithmetik. Das ist in der Tat ein kabbalistischer Kunstgriff, dem eine geniale Intuition zugrunde liegt! Kleene hebt den entscheidenden Gedankenschritt hervor:

›By going over from the conception of the formal system in terms of formal symbols, treated as if they were marks on paper, to an abstract system of objects, our Metamathematics (i. e. the study of the formal system) becomes a branch of pure number theory entirely on a par conceptually with the arithmetic of natural numbers and similar mathematical disciplines.‹²⁸

Gödel selbst gibt die Einsicht, die ihn befähigt, die irreführende Tintenstrichanalogie zu überwinden, in lakonischer Form bekannt:

›Für metamathematische Betrachtungen ist es natürlich gleichgültig, welche Gegenstände man als Grundzeichen nimmt, und wir entschließen uns dazu, natürliche Zahlen als solche zu verwenden. Dementsprechend ist dann eine Formel eine endliche Folge natürlicher Zahlen und eine Beweisfigur eine endliche Folge von endlichen Folgen natürlicher Zahlen. Die metamathematischen Begriffe (Sätze) werden dadurch zu Begriffen (Sätzen) über natürliche Zahlen bzw. Folgen von solchen und daher (wenigstens teilweise) in den Symbolen des Systems PM [Principia Mathematica, R. F.] selbst ausdrückbar.‹²⁹

Die technische Seite des Beweises besteht darin, zu zeigen, daß die metamathematischen Prädikate wie Beweisbarkeit, Nichtbeweisbarkeit, etc. innerhalb der formalen Arithmetik repräsentierbar sind und daß das Nichtbeweisbarkeitsprädikat bzw. die Funktion N von x , die die Gödelnummer

27. Anders als Heintz (a. a. O., S. 150) vermutet, war Gödel kein Formalist. Vgl. Rudy Rucker: *Infinity and the Mind: The Science and Philosophy of the Infinite*. Boston: Birkhäuser, 1982. S. 164–171.

28. Kleene, a. a. O., S. 251.

29. Kurt Gödel: ›Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme‹. *Monatshefte für Mathematik und Physik*, 38 (1931), S. 174.

des Satzes, der aus der Einsetzung des Zahlzeichens für x in das Äquivalent des Nichtbeweisbarkeitsprädikats innerhalb der formalen Arithmetik resultiert, liefert, einen Fixpunkt besitzt, d. h. daß es eine Zahl x_f gibt mit $N(x_f) = x_f$. Der Beweis beruht nicht nur auf der intuitiven Parallelisierung von Beweistheorie und formaler Arithmetik sondern bedient sich auch intuitiver Argumente und – entweder als Zeichen für metamathematische Sachverhalte oder als Abbreviationen formaler Objekte – metamathematischer Symbole. Umso erstaunlicher ist, daß Frau Heintz an mehreren Stellen die Meinung vertritt, ›daß Gödels Beweis ein rein formaler war.‹³⁰ Solche Behauptungen wecken Zweifel daran, ob sie überhaupt versteht, wovon sie redet. Aussagen wie die folgende verstärken diese Zweifel:

›1931, als Gödel seine Resultate publizierte und damit zwei wesentliche Stützen des formalistischen Programms ins Wanken brachte, gab es immerhin noch eine Frage, die vielleicht eine positive Antwort finden würde, die Frage nämlich, ob die Mathematik entscheidbar ist.‹³¹

Genau diese Frage war durch Gödels Publikation, wie schon deren Titel³² nahelegt, bereits negativ entschieden. Gödel hatte die Unvollständigkeit der formalen Arithmetik gezeigt, indem er die Existenz unentscheidbarer Sätze innerhalb derselben nachwies. Die Existenz eines Entscheidungsverfahrens würde nämlich implizieren, daß die wahren Sätze der Arithmetik rekursiv aufzählbar wären und letztere damit auch vollständig wäre.³³

Die ›mechanistische Theorie des Geistes‹³⁴ dank derer die Gleichung mit den Seiten Formalismus und Computerprogramm aufgehen soll, entstammt einem Mißverständnis des Charakters formaler Systeme und ihrer Elemente. Wenn es sich bei den formalen Objekten nur um Markierungen auf dem Papier oder dem Band einer Maschine handelte, dann wäre es nicht einmal möglich zu verifizieren, daß zwei typographische Gebilde dasselbe formale Objekt in verschiedenen Schriften repräsentieren.

Die Auffassung, daß die formalen Objekte nichts mehr bedeuteten³⁵ übernimmt Bettina Heintz von Sybille Krämer, von deren Darstellung der theoretischen Vorgeschichte des Computers³⁶ sie in hohem Maße abhängig ist. Die Grundelemente eines formalen Systems wolle Hilbert

30. Heintz, a. a. O., S. 66 und 283.

31. A. a. O., S. 68

32. ›Über formal *unentscheidbare* Sätze der Principia Mathematica und verwandter Systeme‹

33. Das ist eine Konsequenz aus einem fundamentalen Theorem der Rekursionstheorie, das besagt, daß eine Menge genau dann entscheidbar ist, wenn sie und ihr Komplement rekursiv aufzählbar sind. Vgl. Hartley Rogers, Jr.: *Theory of Recursive Functions and Effective Computability*. Cambridge, MA: MIT Press, 1987, S. 58.

34. Heintz, a. a. O., S. 104.

35. A. a. O., S. 57 und S. 60.

36. Sybille Krämer: *Symbolische Maschinen: Die Idee der Formalisierung im geschichtlichen Abriß*. Darmstadt: Wissenschaftliche Buchgesellschaft, 1988.

- so Frau Krämer - als »»bedeutungslose« Zeichen verstanden wissen.³⁷ Das Wort »bedeutungslos« setzt sie wohl deshalb in Anführungszeichen, weil ihr bei der Sache nicht wohl ist, denn tatsächlich stellt die Formulierung »bedeutungslose Zeichen« eine *Contradictio in adjecto* dar. Ein bedeutungsloses Zeichen ist kein Zeichen!

Tatsächlich hat Hilbert wie jeder Mathematiker ein klares Bewußtsein davon, daß die Elemente einer formalen Theorie sehr wohl Zeichen sind und daß es in der Beweistheorie genauer um ein »rein formale[s] Operieren mit den logischen Zeichen, *das von ihrer Bedeutung ganz absieht* [Hervorhebung von mir R. F.],³⁸ geht. Sollten Hilbert und Ackermann hier von einer Bedeutung absehen, die es nicht gibt?

Natürlich haben die Gebilde formaler Theorien eine Bedeutung: Der Prädikatenlogik, die üblicherweise als Sprache formaler Theorien dient, liegt eine Ontologie zugrunde, von der jeder Mathematiker ein intuitives Verständnis hat: Daß die Strukturen, die Modelle von Theorien darstellen könnten, aus Mengen von unterscheidbaren Objekten bestehen, die bzw. deren cartesische Produkte anhand von Merkmalen unterteilbar sind. Die logischen Partikel (Quantoren und Junktoren) besitzen eine Interpretation, die jeder Mathematiker operationalisieren kann und die logischen Ableitungsregeln sollen korrekt und adäquat sein, d. h. es erlauben, *genau* die semantischen Folgerungen der Axiome abzuleiten. Eine Folgerung zeichnet sich dadurch aus, daß sie immer wahr ist, wenn ihre Prämissen wahr sind. Die Ableitungsregeln müssen also wahrheitserhaltend sein.

Die Aufgabe der mathematischen Semantik besteht darin, Modelle von formalen Theorien zu finden und zu untersuchen. Ein Modell ist die Interpretation der Funktions- und Prädikatenzeichen eines formalen Systems durch die Relationen einer Struktur, die dessen Theoreme in wahre Aussagen verwandelt. Eine widersprüchliche Theorie besitzt kein Modell: Sie ist *bedeutungslos*. Die Behauptung der Widerspruchslosigkeit eines formalen Systems ist eine metamathematische Aussage: daß bestimmte Formeln keine Theoreme des Systems, d. h. innerhalb seiner nicht ableitbar seien.

Die Relevanz der Widerspruchslosigkeit ist rein formal und unter Abstraktion von der Bedeutung der Zeichen überhaupt nicht zu verdeutlichen: Es gibt keinen formalen Grund, weshalb in einem System außer einer Formel ϕ nicht auch die Formel $\neg\phi$ ein Theorem sein sollte. Dagegen sprechen nicht syntaktische sondern nur semantische, inhaltliche Überlegungen, die von unseren ontologischen Überzeugungen ausgehen. Auch Hilbert und Ackermann sehen den intimen Zusammenhang zwi-

37. A. a. O., S. 143.

38. David Hilbert, Wilhelm Ackermann: *Grundzüge der theoretischen Logik*, 6. Aufl., Berlin: Springer, 1972 (Grundlehren der mathematischen Wissenschaften; Bd. xxvii), S. 24 f.

schen Bedeutung und Widerspruchslosigkeit formaler Systeme. Wenn sie, wie es Heintz und Krämer unterstellen, von deren Bedeutungslosigkeit überzeugt wären, bräuchten sie den Widerspruch nicht zu fürchten!

›Um die Frage der Widerspruchsfreiheit in Angriff nehmen zu können, müssen wir erst eine formale Definition des Widerspruchs geben. Gewöhnlich wird darunter verstanden, daß man gleichzeitig zwei Ausdrücke $\mathcal{A}$ und $\neg \mathcal{A}$ herleiten kann. In der Tat wäre das verhängnisvoll, denn da $A \rightarrow (\neg A \rightarrow B)$ eine herleitbare Formel ist, ist auch $\mathcal{A} \rightarrow (\neg \mathcal{A} \rightarrow \mathcal{B})$ eine herleitbare Formel, wo $\mathcal{B}$ ein beliebiger Ausdruck ist. Durch zweimalige Anwendung der Abtrennungsregel würden wir auch die beliebige Formel $\mathcal{B}$ als herleitbare Formel erhalten. Das heißt *der ganze Kalkül wäre zur Bedeutungslosigkeit verurteilt* [Hervorhebung von mir, R.F.], da man in ihm alle Ausdrücke herleiten könnte.³⁹

Wer sich daran stören sollte, daß in einer widersprüchlichen Theorie jede Formel ein Theorem ist, kann natürlich, solange es nur um ein Spiel mit bedeutungslosen Objekten geht, auch die logischen Axiome und Regeln durch beliebige andere ersetzen!

Auch Vollständigkeit ist ein semantischer Begriff. Solange die Formeln einer Theorie nur als bedeutungslose Objekte gelten, ist nicht zu verdeutlichen, weshalb für jede Formel ϕ entweder diese oder ihre Negation $\neg \phi$ ein Theorem sein soll – wobei zu bedenken ist, daß der Junktor $\neg$ unter dieser Prämisse ebenso wenig bedeutet wie irgendein anderes formales Objekt. Der Buchstabe ϕ ist hier kein formales Objekt sondern eine metamathematische Variable, die für eine beliebige Formel steht.

Daß die Durchsetzung der modernen Mathematik die Aufgabe von Inhaltlichkeit und Intuition bedeutete, an deren Stelle schließlich nackte Formalität und mechanisches Operieren träten, ist eine wissenschaftshistorische Fälschung, die dazu dient, einen fatalen Begriff von Rationalität und Technik zu legitimieren. Die beweistheoretischen Formalismen haben durchaus ihre Bedeutung, doch nicht die von Heintz unterstellte:

›Im Gegensatz zu einer weitverbreiteten Ansicht [...] ist die Neuformulierung [von Theorien in einer formalen Sprache R.F.] (die wie jede Beschreibung eines intuitiv erfaßten Gegenstandes wesentlich von unserer Auffassung seiner Natur abhängt) nur ein Hilfsmittel der Grundlagenforschung: es handelt sich nämlich darum, die Bedeutung von Sätzen der informalen Mathematik richtig wiederzugeben und nicht deren syntaktische Struktur; denn der äußeren Form nach haben die formale und die informale Sprache (glücklicherweise!) wenig gemeinsam. Auch sollte man bemerken, daß die durch Umformulierung erzielte Präzision zwar die technische Entwicklung fördert, aber kaum geeignet ist, Schwierigkeiten zu beseitigen, die aus Unzulänglichkeiten der ursprünglichen Begriffe entstehen (gerade das Gegenteil ist der Fall: durch Nachdenken über informale Begriffe werden wir zu einer guten Formalisierung geführt). In den wohlbekannten »Krisen« [...] rührten die Widersprüche von durchaus expliziten Prinzipien (Axiomen, Regeln) her, so daß diese Schwierigkeiten nichts mit ungenügender formaler Präzision zu tun hatten; das Problem lag vielmehr darin, unter verschiedenen, formal präzisen Prinzipien die gültigen herauszufinden. Die eigentliche Grundlagenforschung beschäftigt sich gerade mit Fragestellungen dieser Art,

39. A. a. O., S. 99

die bisweilen ganz andersartige Überlegungen als die übliche Mathematik erfordern. Wir versuchen z. B. vernünftige Gründe *für* die Prinzipien der mathematischen Praxis zu finden, wohingegen in der Mathematik for allem Ableitungen *aus* diesen Prinzipien gesucht werden.⁴⁰

Die Identifikation der Mathematik mit einem mechanisch abzuarbeitenden Formalismus führt tatsächlich sowohl auf wissenschaftspraktische als auch auf wissenschaftstheoretische Schwierigkeiten. Die praktischen sind geeignet, den Mythos der Präzision und Zuverlässigkeit, der die formale Mathematik umgibt, zu zerstören:

»Die formalistische Mathematik, so heißt es erstens, sei besonders zuverlässig. Praktisch ist dies einfach nicht wahr: abstrakte Begriffe und ihre Eigenschaften, die die formalistische Doktrin verwirft, werden in der Mathematik immer wieder dazu verwendet, langwierige Rechnungen zu vermeiden und damit eine bekannte Fehlerquelle (Verrechnen nämlich). Während die größere Durchsichtigkeit abstrakter Argumente allgemein anerkannt wird, übersieht man oft, daß Durchsichtigkeit die praktische Zuverlässigkeit erhöht. Wir wollen noch einmal sehen, was geschieht, wenn wir zu einer abstrakten Definition eine äquivalente formale gefunden haben. Macht die formale Definition den Begriff zuverlässiger? Zum Beispiel, wenn wir formale Regeln zur Erzeugung aller logisch gültigen Formeln gefunden haben? Die tatsächliche, praktische Überzeugung, daß die Konklusion richtig ist, kann doch offenbar nicht von der *Existenz* einer Ableitung abhängen, die wir nicht kennen.«⁴¹

Die Erfahrung, auf die dieses Argument sich bezieht, läßt sich in ähnlicher Form auch außerhalb der Mathematik machen: Formale Ableitungen und Algorithmen sind nicht nur schlechte Werkzeuge der mathematischen Forschung sondern taugen auch nicht besonders gut zum Entwurf und zum Verständnis von Software oder gar von technischen Artefakten. Über den vermeintlichen Tiefsinn des algorithmischen Prinzips ist die Praxis der Technik ebenso hinweggegangen wie die der Mathematik.

Die Vorstellung, daß Mathematik nach Hilbert sich auf die Erzeugung inhaltsloser formaler Gebilde und die Ableitung neuer formaler Gebilde aus gegebenen beschränke, führt nicht nur zu untauglichen Rezepten sondern auch zu einem völlig inadäquaten Verständnis des Erkenntnisfortschritts in der Mathematik.

»Im Hilbertschen Formalismus erzeugt die Mathematik ihre Gegenstände selbst. Es gibt mit anderen Worten keine unabhängige »Außenwelt«, an der die Sätze der Mathematik gemessen werden könnten. Das ist der Grund, weshalb als potentielle Falsifikatoren höchstens logische Widersprüche in Frage kommen, [...]«⁴²

Selbstverständlich gibt es auch in formalen Theorien eine inhaltliche Falsifikation. Z. B. kann ich zeigen, daß die Formel $\forall xy. xy = yx$ kein Theorem der Gruppentheorie ist, indem ich ein Modell, etwa die symmetrische

40. Georg Kreisel, Jean-Louis Krivine: *Modelltheorie: Eine Einführung in die mathematische Logik und Grundagentheorie*, Berlin: Springer, 1972. S. 191 f

41. A. a. O., S. 198

42. Heintz, a. a. O., S. 146

Gruppe S_3 ⁴³ betrachte und durch Nachrechnen zweier Beispiele, etwa von

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}$$

zeige, daß die Formel dort falsch ist. Es wird mir jedoch niemals gelingen, aus jener Formel innerhalb der Gruppentheorie einen Widerspruch abzuleiten, weil auch ihre Negation – was wiederum durch ein Modell, etwa die durch $\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}$ erzeugte Untergruppe von S_3 zu demonstrieren wäre – kein Theorem derselben ist! Vielmehr ist die Gruppentheorie sowohl durch jene Formel als auch ihre Negation konsistent erweiterbar.

Daß es sowohl kommutative als auch nichtkommutative Gruppen gibt, gehört zum mathematischen Elementarwissen. Weniger trivial, doch analog zu der Unterscheidung zwischen kommutativen und nichtkommutativen Gruppen, ist die zwischen euklidischer und nichteuklidischer Geometrie: Die Arbeit der Mathematiker, die schließlich in der Herausbildung nichteuklidischer Geometrien resultierte, bestand hauptsächlich darin, Strukturen zu entdecken, in denen das Parallelenaxiom nicht gilt, ob schon sie Modelle der ersten vier euklidischen Postulate sind. Dadurch war die lange Zeit verfolgte Hypothese widerlegt, daß das fünfte Postulat aus den anderen ableitbar sei. Die Falsifikation dieser Hypothese konnte schon aus logischen Gründen nicht dadurch erfolgen, daß man aus dem Parallelenaxiom einen Widerspruch zu den anderen ableitete, denn jenes ist von diesen ja unabhängig. Die Entdeckung von Modellen, die keine Modelle des Parallelenaxioms sind, spielte für die mathematische Forschung hier eine Schlüsselrolle.⁴⁴ Auch die großen Unabhängigkeitsergebnisse der axiomatischen Mengenlehre wie etwa die Unabhängigkeit des Auswahlaxioms wurden modelltheoretisch bewiesen.⁴⁵

Bereits die oben erwähnten bescheidenen Beispiele verdeutlichen, wie unsinnig – und zwar schon unabhängig von den metamathematischen Resultaten von Gödel, Rosser, Robinson und Church – die Vorstellung ist, Mathematik sei auf die algorithmische Ableitung von Theoremen innerhalb willkürlich konstruierter, uninterpretiert bleibender formaler Systeme reduzierbar. Mathematischer Fortschritt ist das Resultat einer Auseinandersetzung mit Fragen, die sich innerhalb eines konkreten Überlieferungs- und Forschungszusammenhangs stellen. Die Metamathematik hat einerseits die Ausdrucksmöglichkeiten und Werkzeuge, die dabei zur Verfügung stehen, erweitert, andererseits nicht nur den Umfang der Probleme ausgedehnt sondern ihnen eine neue Dimension hinzugefügt. Doch dadurch nimmt sie keine Sonderrolle ein, sondern steht vielmehr in der

43. Das sind die Vertauschungen einer dreielementigen Menge und deren Kombination.

44. Vgl. Jeanne Pfeiffer, Amy Dahan-Dalmedico: *Wege und Irrwege: Eine Geschichte der Mathematik*, Basel: Birkhäuser, 1994, S. 155 ff.

45. eine knappe Darlegung bietet Jerome Malitz: *Introduction to mathematical Logic: Set Theory, Computable Functions, Model Theory*. New York, NY: Springer, 1979, S. 51 ff.

Kontinuität der Mathematikgeschichte. Es ist nicht so, wie die Darstellung von Heintz nahelegt, daß erst der Formalismus die Mathematiker befähigt hätte, zu neuen Begriffsbildungen jenseits einer Welt von intuitiv verbürgten Vorstellungen zu schreiten. Mathematischer Fortschritt involvierte immer schon das Transzendieren von etablierten Vorstellungen. Die Entdeckung der transzendenten und komplexen Zahlen, des Quaternionen- und des Matrizenkalküls, die Herausbildung der Verbands- und der Gruppentheorie, der projektiven Geometrie, etc. – alles Leistungen, die beim Aufkommen der formalistischen Doktrin bereits abgeschlossen waren – nötigten die Mathematiker, etablierte Anschauungen zu überschreiten und abstrakte Begriffe nicht nur zu akzeptieren sondern auch einer erweiterten Intuition zugänglich zu machen.

Die Auffassung, daß es sich bei den mathematischen Theorien, sowie sie in axiomatischem Gewande aufträten, um willkürliche Schöpfungen des menschlichen Geistes handle, ist unhaltbar: Was ein relevantes Problem darstellt und was als Beweis akzeptabel ist, ist nur innerhalb des konkreten Problem- und Diskussionszusammenhangs einer Wissenschaft entscheidbar; es gibt jedoch keine dem Formalismus immanenten Kriterien für mathematische Relevanz und Akzeptabilität.

»However, the definitions given for minimal intuitionistic or classical logic are not arbitrary. They respect some important properties [...] It is not by representing a formalism as an inference system that one does »logic«. There are similar risks in mathematics when one chooses axioms liberally: these have little chance of defining anything of mathematical interest.«⁴⁶

Die Ansicht, daß mathematischer Inhalt verzichtbar sei, ähnelt der, daß Literatur aus Grammatik plus Orthographie bestehe. Der formalistische Beweisbegriff ist epistemisch inadäquat: Zu einem Beweis gehören außer einem gültigen Schluß nämlich auch wahre Prämissen,⁴⁷ die zudem noch gewisse Akzeptabilitätskriterien erfüllen müssen:

»Consider the ordinary notion of a mathematical proof. This is not at all the same as the notion of proof in a formal system (any statement at all can be »proved« in a formal system, by just taking the statement itself, or any statements from which it follows by the rules of the system, as axioms). Nor is it the same as the notion of proof in a system which is sound (one whose axioms are true and whose rules of inference preserve truth). For if Fermat's Last Theorem is in fact true, then a system which has that statement (i. e., the statement »There do not exist positive integers $x, y, z, n > 2$, such that $x^n + y^n = z^n$ «) as its one and only axiom, and any truth-preserving rules of inference you like, is certainly a sound system in which Fermat's Last Theorem can be proved; but this is not what mathematicians call a »proof of Fermat's Last Theorem«. A proof in the ordinary sense (a proof humanly speaking) is a proof in a system which is not just sound, but which a mathematician would be justified in accepting. »Proof« is an *epistemic* notion, not a mathematical one.«⁴⁸

46. René Lalement: *Computation as Logic*. Paris: Masson, 1993, S. 7.

47. Das bemerkt schon Aristoteles, *Analytica posteriora*, Buch i, Kap. 2.

48. Hilary Putnam: *Representation and Reality*. Cambridge, MA: MIT Press, 1988, S. 116 f.

Reale und metaphorische Maschinen

Einer der am meisten mißbrauchten Begriffe der Mathematik bzw. Informatik ist der des Algorithmus. Schon deshalb ist es angezeigt, sich zuerst seiner Bedeutung zu versichern. Wer Algorithmentheorie tatsächlich studiert hat, weiß genau, daß ein Algorithmus ein Verfahren zur Herstellung von *Symbolfolgen* – oder genauer: *formalen Objekten* – aus anderen Gebilden gleicher Art ist. Dies impliziert schon ein informaler und erst recht ein formaler Algorithmenbegriff:

»Roughly speaking, an algorithm is a clerical (i.e., deterministic, bookkeeping) procedure which can be applied to any of a certain class of symbolic *inputs* and which will eventually yield, for each such input, a corresponding symbolic *output*. An example of an algorithm is the usual procedure given in elementary calculus for differentiating polynomials.«⁴⁹

Auch der Duden macht sich diese historisch wohlbegründete⁵⁰ Definition zu eigen und erklärt, daß ein Algorithmus ein »Verfahren zur schrittweisen Umformung von Zeichenreihen« bzw. ein »Rechenvorgang nach einem bestimmten [sich wiederholenden] Schema« sei.⁵¹

Dieser historisch legitimen, Algorithmen als Verfahren des systematischen intellektuellen Umgangs mit Zeichen charakterisierenden Definition steht eine Aufweichung des Begriffs gegenüber, die nicht nur bei Bettina Heintz zu beobachten ist. Sie findet sich auch in der populären und einführenden Informatikliteratur:

»[...] an algorithm is a precise description of a method for solving a particular problem using operations or actions from a wellunderstood repertoire. Algorithms are everywhere. When we change the tires of a car, mow the lawn, vacuum a rug, or follow directions to a restaurant, we use an algorithm. A person can solve problems by means of algorithms expressed in English, provided the person understands English. And a computer can solve problems by means of algorithms expressed in a programming language, provided that computer »understands« that language.«⁵²

Aus einem präzisen Begriff ist hier ein undeutliches Wort für Rezept oder Verfahren zur Lösung einer Aufgabe im weitesten Sinne geworden. Diese Aufweichung des Begriffs hat ihren Ursprung in einigen in die Informatik bzw. das Programmieren einführenden Werken, deren Autoren

49. Rogers, a. a. O., S. 1.

50. Das Wort Algorithmus leitet sich aus dem Namen des arabischen Gelehrten al Hwarizmī ab, der im 9. Jahrhundert mehrere einflußreiche Werke über das Rechnen mit indischen Ziffern – die nicht zuletzt deshalb heute arabische heißen – verfaßte. Vgl. Paul Benoît, Françoise Micheau: »Die Araber als Vermittler?«. Michel Serres (Hrsg.): *Elemente einer Geschichte der Wissenschaften*. Frankfurt am Main: Suhrkamp, 1994, S. 297 ff.

51. Duden: *Das große Wörterbuch der deutschen Sprache*. Mannheim: Dudenverlag, 1993.

52. John Shore: *The Sachertorte Algorithm and Other Antidotes to Computer Anxiety*. New York, NY: Penguin, 1986, S. 131.

durch das gutgemeinte pädagogische Ansinnen, diese abstrakte Materie mittels konkreter Bilder zu illustrieren, motiviert waren:

»A fundamental concept in computer science is that of an algorithm. Informally, we can view an algorithm simply as a way to solve a problem, or a set of directions, or a recipe that tells exactly how to go about getting desired results. We work with algorithms all the time, although we are not often so formal to call them algorithms. The set of instructions that tells us how to put a tricycle together from its component parts, the procedure for going through college registration, and the recipe for making an apple pie are all probably examples of valid algorithms. Even the following directions taken from a shampoo label could be called a »shampooing algorithm«.⁵³

Gut gemeint ist bekanntlich oft das Gegenteil von gut. Der ehrenwerte Vorsatz, die Studenten zu unterhalten indem man ihnen die Merkmale von Algorithmen anhand eines »shampooing algorithm« erläutert, zwang hier zur Verwässerung des Algorithmenbegriffs – die Gültigkeit des Beispiels requiriert schließlich dessen Subsumtion unter den Begriff des Algorithmus – und setzte damit eine Kette von Mißverständnissen in Gang, an deren Ende anmaßende theoretische Konstruktionen wie die des panalgorithmischen Programms stehen, deren Urheber ihr Informatikverständnis anscheinend vor allem aus Literatur der referierten Art beziehen.

Die falsche Identifikation von Intellekt, Algorithmus und Maschine, die den Kern des panalgorithmischen Programms ausmacht, scheitert an drei schlichten Sachverhalten:

1. Der algorithmische ist nur ein untergeordneter Modus des Intellekts.
2. Kochrezepte, technische Verfahren und Funktionsweisen von Maschinen sind nur im metaphorischen Sinne Algorithmen.
3. Algorithmen sind nur im metaphorischen Sinne Maschinen.

Aus diesen Sachverhalten folgt weiter, daß Computer *spezielle, nicht jedoch universelle* Maschinen sind. Den ersten hat bereits der vorige Abschnitt dargetan.

Algorithmen sind wie Rezepte, Verfahren, etc. *geistige* Gebilde. Doch während ein Rezept oder ein technisches Verfahren eine Anleitung zur Herstellung eines *materiellen* Produkts aus *materiellen* Rohstoffen ist, der Aufbau einer Maschine die Konfiguration *physischer* Komponenten und ihre Funktionsweise den Zusammenhang ihrer *physischen* Zustände, Ein- und Ausgaben bestimmt, ist ein Algorithmus eine Vorschrift zur Bildung eines geistigen Resultats aus geistigen Ingredienzen.

Ein Algorithmen ist *kein* Verfahren zur Umformung, zum Transport oder zur Speicherung von Materie und Energie. Technische Verfahren, Maschinen, Vorrichtungen, etc. zielen darauf, einen *physischen* Effekt zu erzielen, Algorithmen jedoch nicht. Die Ausführung eines Algorithmus durch einen menschlichen Intellekt impliziert zwar auch physische

53. G. Michael Schneider, Steven W. Weingart, David M. Perlman: *An Introduction to Programming and Problem Solving with Pascal*. New York, NY: Wiley, 1978, S. 19.

Auswirkungen – das Erregungsniveau von Neuronen ändert sich, Papier füllt sich mit Tintenmarken, Sauerstoff wird verbraucht, Abacusperlen verändern ihre Position, etc. – doch gehören diese nicht zu den *intendierten* oder gar *charakteristischen* Effekten dieser Tätigkeit. Ganz anders bei der Anwendung von stofflichen Verfahren und Maschinen: Wenn ich Kaffee male, einen Stamm in Bretter zersäge oder Eisenbahn fahre, dann um des jeweiligen *physischen* Effekts willen.

Um Mißverständnisse auszuschließen: Das bedeutet nicht, daß z.B. eine Tätigkeit wie Kaffeemahlen oder eine Reise mit der Eisenbahn nicht auch Zeichen sein könnten: ich kann damit Abneigung, Anteilnahme, Zuneigung, Liebe, etc. ausdrücken, je nachdem in welchem Kontext ich Kaffee mahle oder wann, von wo, wohin ich reise. Doch das gehört nicht zur Beschreibung des Kaffeemahlens oder der Eisenbahn. Auch müssen Maschinen, Vorrichtungen, als Zeichen ihrer Funktion funktionieren, um diese im Kontext menschlichen Handelns annehmen zu können.⁵⁴ Doch reicht es nicht aus, daß ein technisches Artefakt seine Funktion denotiert: es muß sie auch physisch erbringen. Auf einer Treppe kann ich genau deshalb in ein anderes Geschoß steigen, weil sie bestimmte *physikalische* Kriterien erfüllt, also das Hinaufsteigen nicht nur denotiert sondern auch physisch ermöglicht. Liebhaber des Heintzschen Technikbegriffs mögen an dieser Stelle eine kurze Pause einlegen und darüber nachdenken, welchen Algorithmus die Treppe dabei ausführt.

Eine Intention auf der symbolischen Ebene kann ganz bestimmte physische Bedingungen spezifizieren: Wer z.B. seiner Erbtante Zuneigung signalisieren möchte, tut gut daran, pünktlich zu ihrer Geburtstagseinladung zu erscheinen und ihre Lieblingsblumen dabei zu haben. Noch konsequenter tritt dieses Prinzip bei Computern hervor: Ein Computer ist eine *spezielle* Maschine, die dazu konstruiert ist, auf bestimmte *physikalisch spezifizierte Eingaben* mit bestimmten *physikalisch determinierten Zustandsänderungen und Ausgaben* zu antworten!

Daß Zustände, Ein- und Ausgaben eines Rechners dank seiner besonderen Konstruktion so beschaffen sind, daß sie als Träger für ein Zeichensystem zu fungieren vermögen, macht aus ihm eine Maschine zur Realisierung von Elementen und Transformationen dieses Zeichensystems innerhalb des eng umgrenzten Systems seiner physikalisch spezifizierten Zustände, Ein- und Ausgaben. Wenn ein Computer rechnet, durchläuft er eine Folge von Zuständen, *die durch bestimmte Werte physikalischer Größen spezifiziert sind und deren Abfolge physikalische Gesetze determinieren*. Diese Abfolge reflektiert nur innerhalb bestimmter physikalisch spezifizierter Randbedingungen die intendierten Transformationen des Zeichensystems. Zu hohe oder zu niedere Temperaturen, starke elektromagnetische Felder, intensive energiereiche Strahlung, etc. sind geeignet,

54. Vgl. Umberto Eco: *Einführung in die Semiotik*. München: Wilhelm Fink, 1972. S. 306 ff.

einen Rechner zu Zustandsübergängen zu veranlassen, die keinen intendierten Transformationen des Zeichensystems entsprechen.

Universell ist ein Rechner in sofern, als er – innerhalb gewisser Grenzen, die der verfügbare Speicher und die bereitstehende Zeit setzen – jeden Algorithmus auf dem Zeichensystem ausführen kann, und dadurch, daß Zeichensysteme und Algorithmen bzw. deren Denotationen als Material zur Modellbildung geeignet sind. Der Computer ist keine universelle Maschine im strengen Sinne – mit einem Computer kann man beinahe nichts von dem tun, was man mit anderen Maschinen tun kann – sondern eine universelle algorithmische Maschine und eine universelle Modellmaschine.⁵⁵ Eine universelle algorithmische Maschine ist also eine *spezielle* Maschine, die dazu in der Lage ist, auf einer bestimmten physischen Realisierung eines Zeichensystems innerhalb der gegebenen Grenzen von Speicherplatz und Zeit jeden Algorithmus auszuführen. Allerdings führt der Rechner algorithmische Operationen nicht intentional als Zeichenoperationen aus, sondern indem er physikalisch determinierte Zustandsübergänge erfährt.

Computer sind physische Gebilde, die denselben physikalischen Gesetzen unterliegen wie andere Maschinen. Ihre Eigenschaft als Realisationen von Zeichensystemen verdanken sie den Intentionen ihrer Schöpfer und Benutzer und sie erfüllen diese Intentionen, indem sie spezifischen physikalischen Bedingungen entsprechen.

Auch Maschinen, deren primärer Zweck darin besteht, physische Effekte zu erzielen, sind als Realisierungen von Zeichensystemen interpretierbar. Die Gewinde, die eine Drehmaschine produziert, codieren eine Botschaft und jene Maschine umfaßt Funktionen der Informationsübertragung, -speicherung und -verarbeitung. Umgekehrt ist ein Computer auch eine Heizung. Es ist nicht so, daß die Artefakte in zwei disjunkte Klassen zerfallen würden – klassische, auf physische Effekte hin konstruierte Maschinen auf der einen, physisch realisierte Zeichensysteme auf der anderen Seite. Vielmehr sind dies intentionale Aspekte, die im Prinzip an jedem Artefakt hervortreten. Schon eine Postkutsche war zugleich ein Transport- und ein Kommunikationsmittel; eine Kupferleitung transportiert Energie und übermittelt Zeichen, etc. Doch gerade das Übermitteln, Speichern und Manipulieren von Zeichen geschieht in den informationstechnischen Maschinen stets dadurch, daß ein genau spezifizierter physischer Effekt auftritt. Dessen intentionale Deutung als Zeichen ist Sache des Anwenders.

Algorithmen sind dadurch gekennzeichnet, daß man sie geistig ausführen kann – wobei die physischen Seiteneffekte keine essentielle Bedeutung haben – und daß ihr Resultat auch nur geistig auffaßbar ist. Der Zweck eines Algorithmus impliziert nur die Bewegung von geistigem In-

55. Vgl. Fischbach, a. a. O.

halt, nicht die von Materie und Energie; auch wenn seine Ausführung dies als Seiteneffekt involviert. Die Bedeutung von Algorithmen ist deshalb auch ohne Bezugnahme auf empirische Sachverhalte, physikalische Größen und Gesetze beschreibbar.

Die formalen algorithmischen Sprachen legen verschiedene semantische Interpretationen nahe: Den Phrasen mancher dieser Sprachen lassen sich Werte eines mathematischen Bereichs als Denotationen zuordnen, die anderer sind als Spezifikationen solcher Werte interpretierbar. Die Semantik muß entweder ein Evaluationsverfahren für die Phrasen angeben oder ein Verfahren, das es erlaubt, einen Wert und einen Beweis dafür zu finden, daß er die Spezifikation erfüllt. Dabei findet meist eine um semantischen Axiome für die algorithmische Sprache erweiterte Form des auf Gentzen zurückgehenden Kalküls des natürlichen Schließens⁵⁶ Anwendung. Den intuitionistischen Varianten dieses Kalküls kommt hier besondere Bedeutung zu. Die klassische Schlußregel [K], die es erlaubt, aus dem Sachverhalt, daß die Annahme der Nichtexistenz eines Wertes, der eine Spezifikation erfüllt, zu einem Widerspruch $\perp$ führt, auf dessen Existenz zu schließen, hat keinen konstruktiven Sinn:

$$[\neg \exists \alpha. \phi]$$

$$\frac{\vdots}{\perp} \quad \frac{\phi[\sigma/\alpha]}{\exists \alpha. \phi} \quad \frac{}{\exists \alpha. \phi} [K] \quad \frac{}{\exists \alpha. \phi} [I]$$

Ein solcher reiner Existenzbeweis gibt kein Mittel an die Hand, einen Wert zu finden, der die Formel ϕ erfüllt. Die intuitionistische Logik, die fordert, daß ein solcher Wert σ vorzulegen sei, um den Schluß auf die Existenz zu rechtfertigen, läßt die Regel [K] nicht zu. Die Alternative formalisiert die intuitionistische Regel [I], die mit mehr Recht in der semantischen Logik einer algorithmischen Sprache einen Platz beanspruchen kann.

Auch hier liegt wiederum ein Sachverhalt vor, den es nicht geben dürfte, wenn das von Bettina Heintz gezeichnete Bild der Mathematik und Informatik korrekt wäre. Für Heintz bedeutet der Aufstieg des Formalismus den Abstieg und das Verschwinden des Intuitionismus. Doch tatsächlich gibt es seit Jahrzehnten bedeutende intuitionistische Beiträge zur Mathematik und vor allem die Problemstellungen der Semantik von Programmiersprachen haben dem Intuitionismus in den letzten Jahren starken Auftrieb gegeben.⁵⁷

Der Sinn einer formalen Semantik besteht darin, einen präzisen Maßstab für die die Bewertung der Implementation algorithmischer Sprachen

56. Vgl. G. Gentzen: ›Untersuchungen über das logische Schließen‹. *Mathematische Zeitschrift*, 39 (1934–1935), S. 176–210 u. S. 405–431 sowie Tennant, a. a. O.

57. Es gibt zu diesem Thema eine Vielzahl von recht anspruchsvollen Veröffentlichungen. Eine vergleichsweise einfache Einführung in das Gebiet bietet Simon Thompson: *Type Theory and Functional Programming*. Wokingham: Addison-Wesley, 1991.

zu liefern und eine theoretische Untersuchung der Eigenschaften von Programmen zu ermöglichen. Neben dem Beweis, daß ein Programm seine Spezifikation erfüllt, spielt hierbei häufig die Frage eine Rolle, ob zwei verschiedene Programme äquivalent sind in dem Sinne, daß sie für identische Eingaben auch identische Ausgaben produzieren. Ich komme auf dieses – algorithmisch unentscheidbare – Problem später zurück.

Die Regeln natürlicher Semantiken führen – in der Tradition des Kalküls des natürlichen Schließens – die Prämissen $\phi_1 \dots \phi_n$ und die Konklusion ψ in folgender Gestalt auf:

$$\frac{\phi_1 \dots \phi_n}{\psi}$$

Die Regeln sind metasprachliche Figuren, die unendliche Klassen von Schlüssen denotieren. Ein Beweis in der Objektsprache hat die Gestalt eines Baumes, dessen Blätter Axiome oder eliminierte Hypothesen sind, dessen innere Knoten aus Regelanwendungen bestehen und dessen Wurzel das Theorem bildet. Der Beweis von $R(a)$ aus $\forall x. P(x) \wedge Q(x) \rightarrow R(x)$, $P(a)$ und $Q(a)$ unter Anwendung der Partikularisierungsregel [Part], der $\wedge$ -Einführung [$\wedge$ Intro] und des Modus ponens [MP] illustriert dies:

$$\frac{\frac{\forall x. P(x) \wedge Q(x) \rightarrow R(x)}{P(a) \wedge Q(a) \rightarrow R(a)} [\text{Part}] \quad \frac{P(a) \quad Q(a)}{P(a) \wedge Q(a)} [\wedge \text{Intro}]}{R(a)} [\text{MP}]$$

In einem solchen Beweisbaum gibt es *keine zeitliche Abfolge*! Anders als Heintz vermutet,⁵⁸ zwingen die Regeln des logischen Kalküls nicht dazu, die einzelnen Inferenzen in einer bestimmten Folge zu vollziehen bzw. deren Prämissen nach einer bestimmten Ordnung zu beweisen. Der Mathematiker, der in einem Kalkül arbeitet, kann seine Strategie nach Belieben wählen und der Informatiker, der ihn implementiert, hat die Auswahl zwischen verschiedenen technischen Alternativen!

Die These vom ›Fließband im Kopf‹⁵⁹ steht auf recht schwachen Füßen! Beweise in formalen Systemen sind nicht algorithmisch in dem von Heintz gemeinten Sinne einer sequentiell verfahrenen mechanischen Prozedur! Bettina Heintz erliegt der von dem Falsifikat ›formalistische Mathematik‹ ausgehenden Faszination, wenn sie glaubt, feststellen zu können:

›Zur selben Zeit, als man in der Soziologie den Modernisierungsprozeß systematisch unter dem Blickwinkel der (formalen) Rationalisierung zu deuten begann, setzte sich in der Mathematik die Auffassung durch, daß man das Betreiben von Mathematik reduzieren könne auf einen rein formalen Prozeß, der im wesentlichen darin bestehe, Zeichenreihen Schritt für Schritt umzuformen, nach klaren Regeln und ohne Ansehung ihrer Bedeutung.‹⁶⁰

58. Heintz, a. a. O., S. 234.

59. A. a. O., S. 154 ff., S. 176 und S. 172.

60. A. a. O., S. 164.

Anschaung reduziere sich bei diesem rein mechanischen Prozeß auf die ›Wahrnehmung von Markierungen auf dem Papier‹.⁶¹ Die Mathematik werde zur Industrie:

›Im Hilbertschen Formalismus wird die Mathematik umgebaut zu einem Regelwerk, zu einer Maschinerie, in der der Mathematiker – sind die Axiome einmal gesetzt – im Prinzip nur noch Vollzugsfunktion hat.‹⁶²

In Turings und Posts theoretischen Konstrukten gerinne schließlich das neue Modell der mathematischen Arbeit zu konkreter Gestalt:

›Ein unendliches Band. Unterteilung in Felder. Einfachste Handlungen. Mechanisches Ausführen von Befehlen. Schrittweises Vorgehen. Sequentielle Anordnung. Der Grundgedanke blieb sich bei beiden gleich: Das Befolgen eines Algorithmus ist ein Prozeß, dessen Ausführung keine Abweichung, keinen Spielraum zuläßt.‹⁶³

Nicht nur, daß Heintz den Stellenwert formaler Systeme innerhalb der Mathematik völlig falsch einschätzt: Sie versteht nicht einmal ihren Charakter. Formale Beweise sind ideale, zeitlose mathematische Objekte, in denen es keine zwingene Abfolge von Schritten gibt. Das Modell von Turing, das nichts anderes ist als die abstrakte Imitationen eines Menschen, der in ziemlich phantasieloser Weise mit Papier, Bleistift und Radiergummi rechnet, enthält gerade in seiner Bildhaftigkeit ein überschüssiges, durch den theoretischen Gehalt der Sache nicht gerechtfertigtes Maß an Rigidität. Nicht zuletzt mag dies – neben all den außertheoretischen Determinanten und Kontingenzen der Wissenschaftsgeschichte – ein weiterer Grund für die Unfruchtbarkeit dieses Konstrukts gewesen sein.

Daß Turing ›mit seinem Maschinenmodell das Grunddesign von Digitalcomputern vorweggenommen‹⁶⁴ habe, ist völliger Unsinn. Seine physikalischen Prinzipien sind nämlich – weshalb es keine wirkliche sondern nur eine metaphorische Maschine beschreibt – nicht nur völlig unbestimmt oder – wie im Fall des unendlichen Bandes – nichtexistent, sondern seine logischen sind so stark durch seine wenig elegante Architektur und Funktionsweise geprägt, daß es sich als Vorbild weder für den Entwurf realer Computer noch für den von Programmiersprachen eignet.⁶⁵

Nur die Ableitungen innerhalb bestimmter formaler Systeme eignen sich als Modelle für Berechnungen. Wesentlich für diese Systeme ist, daß sie ein inhaltliches Kriterium erfüllen: Sie müssen konstruktiv im Sinne des Intuitionismus sein. Wobei zu bemerken ist, daß jede dieser Ableitungen ein ideales mathematisches Objekt ist. Die Theorie der Algorithmen

61. A. a. O.

62. A. a. O., S. 165.

63. A. a. O., S. 167.

64. A. a. O., S. 106.

65. Wie gering der Einfluß der Turingmaschine auf die tatsächliche Geschichte des Computers war, ist im einzelnen nachzulesen bei Pierre Lévy: ›Die Erfindung des Computers‹. Serres, a. a. O., S. 905–944.

handelt von unendlichen Klassen solcher Objekte. Einen geeigneten Formalismus bildet z. B. die Logik der Hornklauseln. Berechnungen sind Beweise in dieser Logik. Die Programmiersprache Prolog erlaubt die Formulierung solcher Klauseln und eine Implementation dieser Sprache ist – manchmal – dazu in der Lage, Beweise oder Widerlegungen für Formeln in dieser Logik zu finden. Die Axiome des vorherigen Beweisbeispiels lauten in Prolog:

```
r(X) :- p(X), q(X).
p(a).
q(a).
```

Prolog verwendet nur eine Ableitungsregel, den verallgemeinerten Modus ponens.⁶⁶ Eine Anfrage in Form des Zieles $\text{?- } r(a)$ liefert tatsächlich die Antwort yes . Würde man nur $\text{?- } r(X)$ fragen – also: $\text{gibt s ein Objekt } X$, auf das r zutrifft?, wäre das Suchverfahren sogar dazu in der Lage, dieses zu finden und würde mit $X = a$ antworten.

Die Sequentialisierung der Beweisschritte, die übliche Prolog-Implementationen vornehmen, führt in manchen Fällen dazu, daß die Suche nach einem Beweis nicht terminiert, obwohl ein solcher für das angegebene Ziel existiert. Prolog-Interpreter sind rückwärtsverkettende Beweiser, die überprüfen, ob eine Formel ein Theorem ist bzw. ob es Belegungen ihrer Variablen gibt, die sie zu einem Theorem machen, indem sie versuchen, von dieser Formel als Wurzel ausgehend einen Beweisbaum – also einen solchen, dessen Blätter Axiome und dessen innere Knoten Regelanwendungen sind – zu konstruieren. Die meisten dieser Beweiser führen dabei eine Tiefensuche aus, die alle potentiellen Äste des Beweisbaums exploriert. Diese ist inhärent sequentiell. Da manche Äste unendlich sind, kann es zu nichtterminierenden Berechnungen kommen.

Die Nutzung von Parallelrechnern verlangt dagegen eine Breitensuche, die im Gegensatz zur Tiefensuche keine strikte Sequentialisierung bedingt. Die Breitensuche hat zudem den Vorzug, daß sie Beweise entdeckt, die die Tiefensuche niemals erreicht, wenn sie in einen unendlichen Ast gerät. Eine im Vergleich zur herkömmlichen Technik effizientere und effektivere Implementation auf Parallelrechnern ist möglich, weil der zugrunde liegende Formalismus algorithmisch weitgehend unbestimmt ist. Die tatsächliche Abfolge der Schritte ist beweistheoretisch ohne Bedeutung, wobei ein menschlicher Prozessor auch zwischen den Ästen des Baumes springen und metasprachliche Abkürzungen verwenden wird, um sich die Arbeit zu vereinfachen.

66. Dieser schließt neben der Partikularisierung, die hier in Form der Unifikation von Termen auftritt, auch die $\wedge$ -Einführung ein. Eine genauere Diskussion bietet Leon Sterling, Ehud Shapiro: *The Art of Prolog: Advanced Programming Techniques*. 2. Aufl., Cambridge, MA: MIT Press, 1994.

Aus gutem Grund vermeidet man es, formale Systeme – vor allem auch Softwaresysteme – durch Algorithmen in dem von Heintz gemeinten Sinne zu beschreiben: Solche Beschreibungen sind nur sehr schwer verständlich und legen der Implementation zu enge Fesseln an. Vor allem sequentielle Algorithmen besitzen deshalb sowohl aus kognitiven als auch aus technischen Gründen weder in der formalen Mathematik noch in der Informatik die Bedeutung, die Heintz ihnen zuschreibt!

Hier ist der Ort, um eine Bemerkung aus dem vorigen Abschnitt technisch zu vertiefen: Ein formaler Beweis ist ein ideales mathematisches Objekt. Er ist weder identisch mit der Ausführung einer sequentiellen mechanischen Prozedur noch mit einem typographischen Gebilde, d. h. einer Anordnung von physischen Marken auf Papier. Vielmehr entspricht jedem Beweisobjekt eine potentiell unendliche – daß es auch solche Beweisobjekte gibt, die hinzuschreiben die Materie im uns bekannten Teil des Universums nicht ausreichen würde, habe ich bereits erwähnt – Klasse solcher Gebilde. Selbst die idealisierten typographischen Gebilde sind nur Repräsentanten von Äquivalenzklassen alphabetischer Varianten: Im Beispiel ist es ohne Bedeutung, wenn man $\forall x. P(x) \wedge Q(x) \rightarrow R(x)$ durch die Variante $\forall y. P(y) \wedge Q(y) \rightarrow R(y)$ ersetzt – was zulässig ist, sofern die Formeln gewisse syntaktische Bedingungen erfüllen. Beide sind äquivalent – ein Hinweis darauf, daß selbst die Arbeit in formalen Systemen etwas anderes ist als das mechanische Hantieren mit Marken auf Papier! Die Implementation solcher Systeme verlangt die Auswahl von Repräsentationen und Operationssequenzen. Letzten Endes muß dies zur Spezifikation von physischen Maschinenzuständen und Zustandsübergängen führen, doch ist dies mathematisch bedeutungslos und selbst die Informatiker interessiert das meist nicht besonders.

Maschinell zu rechnen bedeutet, idealen syntaktischen Strukturen, die ideale mathematische Objekte denotieren, eine konkrete physische Darstellung im Speicher eines Rechners zu geben und die Resultate der durch physikalische Gesetze determinierten Zustandsübergänge dieses Speichers ebenfalls als Repräsentanten idealer syntaktischer Strukturen zu interpretieren, die wiederum für ideale mathematische Objekte stehen. Die Kunst des Rechnerbaus besteht darin, genau solche Zustandsübergänge zu erlauben, die kongruent sind mit einer bestimmten Klasse von syntaktischen Transformationen.

Algorithmisches Rechnen besteht darin, Zeichen zu transformieren: Etwa das Zeichen $\succ 6 * 9 \prec$ in das Zeichen $\succ 54 \prec$. Den dazu erforderlichen Algorithmus haben Angehörige des westlichen Kulturkreises meistens in der Grundschule gelernt. Die Algorithmen befähigen dazu, auch mit unvorstellbaren Zahlen zu rechnen, indem sie das Operieren mit Zahlen durch das Operieren mit Ziffern, d. h. Zahlzeichen ersetzen. Darin liegt die Rationalisierungsleistung der indischen und arabischen Rechentechnik.

nik. Die Rechenmaschinen ersetzen schließlich das mentale Operieren mit Zeichen durch das Operieren mit physischen Objekten bzw. deren Zuständen. Als Lösung einer Rechenaufgabe erscheint das jedoch nur für jemanden, der die Aufgabe versteht und dazu in der Lage ist, das Ergebnis zu interpretieren.

Eine denotationelle Semantik ordnet den syntaktischen Gebilden einer Sprache – z.B. der Sprache der arithmetischen Terme – mathematische Objekte als Bedeutungen zu. Sie verfährt dabei kompositionell, d. h., sie definiert die Bedeutung einer komplexen Phrase als Komposition aus den Bedeutungen ihrer Komponenten. Im obigen Beispiel heißt das, daß die Bedeutung von $\langle 6 * 9 \rangle$ sich aus der Bedeutung von $\langle 6 \rangle$ von $\langle * \rangle$ und von $\langle 9 \rangle$ ergibt. Die Implementation einer algorithmischen Sprache muß für deren Terme einen *Evaluator* bereitstellen, der die Spezifikationen der Semantik erfüllt. Da der Evaluator eine syntaktische Transformation ausführt, liefert er nicht unmittelbar das mathematische Objekt, das ein Term denotiert, sondern den Term, der dessen *kanonische* Bezeichnung ist. Dieser repräsentiert eine Klasse äquivalenter Terme.

Eine denotationelle Semantik teilt folglich die Terme der Sprache in Äquivalenzklassen ein: alle Mitglieder einer solchen Klasse denotieren dasselbe mathematische Objekt. Der Evaluator reduziert die Terme einer Klasse auf ihren kanonischen Repräsentanten, z.B. $\langle 6 * 9 \rangle$ und $\langle 9 * 6 \rangle$ auf $\langle 54 \rangle$. Alle diese Terme denotieren die Zahl 54. Ein besonderes Problem stellen Terme dar, deren Evaluation nicht terminiert. Auch ihnen kann eine denotationelle Semantik einen Wert zuordnen. Doch das soll uns hier nicht weiter beschäftigen.

Ein anderes schwer zu behandelndes Problem bilden Ausdrücke, die funktionale Abstraktionen darstellen. Eine funktionale Abstraktion ist die Verallgemeinerung eines Terms. Während etwa $\langle 6 * 9 \rangle$ eine ganze Zahl bezeichnet, steht $\langle \text{fn } x \Rightarrow 6 * x \rangle$ für die Funktion, die jeder ganzen Zahl ihr Sechsfaches zuordnet. Für jedes Zahlzeichen ζ ⁶⁷ liefert die Anwendung dieser Abstraktion, also der Term $\langle (\text{fn } x \Rightarrow 6 * x) \zeta \rangle$, den ganzzahligen Term $\langle 6 * \zeta \rangle$, der nach den Regeln für solche Terme weiter zu evaluieren ist. Während z.B. alle Terme, die die ganze Zahl 54 denotieren, zum Term $\langle 54 \rangle$ evaluieren, kann jedoch kein Evaluator alle funktionalen Abstraktionen, die dieselbe Funktion denotieren, zu demselben Term reduzieren. Die Abstraktion $\langle \text{fn } x \Rightarrow 69 * 6 / 69 * x \rangle$ denotiert dasselbe mathematische Objekt wie $\langle \text{fn } x \Rightarrow 6 * x \rangle$; doch werden die meisten Evaluatoren diese Äquivalenz nicht erkennen, d. h. der Ausdruck

$$(\text{fn } x \Rightarrow 6 * x) = (\text{fn } x \Rightarrow 69 * 6 / 69 * x)$$

67. ζ ist eine metasprachliche Variable, die also nicht der betrachteten algorithmischen Sprache sondern der Sprache, in der von jener Sprache die Rede ist, angehört.

evaluiert in manchen algorithmischen Sprachen zu `>false<`, in anderen – wie in der hier verwendeten SML – gilt er als unzulässig, weil die Werte von Abstraktionen dort nicht vergleichbar sind. Andererseits sind die beiden Abstraktionen *extensional äquivalent*; d. h. für jedes Zahlzeichen ζ wird

$$(\text{fn } x \Rightarrow 6 * x) \zeta = (\text{fn } x \Rightarrow 69 * 6 / 69 * x) \zeta$$

zu `>true<` evaluieren und die Semantik muß beiden Abstraktionen dieselbe Funktion als Bedeutung zuordnen. Einen Evaluator, der alle extensional äquivalenten Abstraktionen erkennt, gibt es jedoch nicht. Seine Existenz implizierte die algorithmische Lösbarkeit des Halteproblems.

Selbstverständlich ist die extensionale Äquivalenz von Abstraktionen von hoher praktischer Bedeutung und ungeachtet ihrer algorithmischen Unentscheidbarkeit gelingt es manchmal, die eine oder andere Äquivalenz innerhalb einer semantischen Theorie algorithmischer Sprachen zu beweisen oder zu widerlegen. Solche Beweise bedürfen meist arithmetischer oder äquivalenter Mittel wie der strukturellen oder wohlfundierten Induktion. Worin liegt die Bedeutung der extensionalen Äquivalenz von Abstraktionen?

1. Oft besitzen wir für die Berechnung eines Ergebnisses zunächst kein Verfahren sondern nur eine *Spezifikation*, die dessen Eigenschaften beschreibt. Gesucht ist ein Verfahren, das die Spezifikation erfüllt.
2. Nicht selten fällt uns zwar spontan ein Verfahren ein, doch ist dieses nicht effizient. Wir suchen dann nach einem Verfahren, das dieselbe Spezifikation erfüllt, das also, sofern diese eindeutig eine Funktion bestimmt, extensional äquivalent mit dem ersten ist.

Ein klassisches Beispiel stellen die Fibonacci-Zahlen dar. In einer Sprache wie SML ist deren Definition ausführbar:

```
val rec fibonacci =
  fn 0 => 0
  | 1 => 1
  | n => fibonacci (n - 1) + fibonacci (n - 2);
```

Die funktionale Abstraktion, die hier an den Namen `fibonacci` gebunden ist, ist als unmittelbare programmiersprachliche Übersetzung der Definition der Fibonacci-Zahlen erkenntlich. Leider ist sie sehr ineffizient. Ihr Zeit und Speicherbedarf wachsen exponentiell mit dem Wert des Arguments. Die beiden folgenden Abstraktionen sind mit der obigen extensional äquivalent, berechnen also ebenfalls die Fibonacci-Zahlen, benötigen jedoch – sofern man wie die meisten Implementationen von Programmiersprachen mit Zahldarstellungen fester Breite, die den Bereich der repräsentierbaren Werte einschränken, arbeitet und der Übersetzer

Endrekursion eliminieren kann - nur eine linear mit dem Argument wachsende Rechenzeit und konstanten Speicher!

```

val fibonacci1 =
  let
    val rec ft =
      fn (0, u, v) => u
      | (1, u, v) => v
      | (n, u, v) => ft (n - 1, v, u + v);
    in
      fn m => ft (m, 0, 1)
    end;
  val fibonacci2 =
    let
      val sqrt5 = sqrt 5.0;
      val phi = (1.0 + sqrt5) / 2.0;
      val phi' = (1.0 - sqrt5) / 2.0;
      val round = fn x => floor (x + 0.5);
      val exp = fn (x, k) =>
        let
          val rec pow =
            fn (0, y) => y
            | (n, z) => pow (n - 1, x * z);
          in
            pow (k, 1.0)
          end;
        in
          fn m =>
            round ((exp (phi, m) - exp (phi', m)) / sqrt5)
        end;
    end;

```

Viele Eigenschaften dieser Abstraktionen - ihre mathematische Denotation, ihr Zeit- und Speicherverhalten - sind innerhalb der Semantik Konsequenzen ihrer syntaktischen Gestalt; wobei algorithmische Details wie die Folge, in der die Komponenten eines Ausdrucks evaluiert werden, unbestimmt bleiben. Allerdings verlangt es eine gewisse Anstrengung, diese Eigenschaften aus der syntaktischen Struktur abzuleiten und z.B. den Beweis zu führen, daß zwei recht verschiedene syntaktische Gebilde hinsichtlich einiger Eigenschaften übereinstimmen. Im vorliegenden Fall ist der Beweis der semantischen Äquivalenzen durch vollständige Induktion über den Parameter der Abstraktionen zu führen.⁶⁸

68. Eine Einführung in die Technik des semantischen Beweises auf der Basis von SML bringt Laurence C. Paulson: *ML for the Working Programmer*. Cambridge: Cambridge University Press, 1991, S.187 ff.

Eine Folgerung aus der Einsicht, daß verschiedene Algorithmen dasselbe Verhalten zeigen können, ist ebenso schlicht wie für die zentrale These von Heintz vernichtend: Algorithmen sind *nicht entscheidend* für das Verhalten von Berechnungen, geschweige denn für das von Maschinen. *Sie charakterisieren es nicht*: Ein bestimmtes Verhalten muß nicht die Folge eines bestimmten Algorithmus sein; vielmehr kann es durch verschiedene Algorithmen verursacht sein.

Die Klasse der berechenbaren Funktionen – also derjenigen mathematischen Objekte, die eine denotationelle Semantik den funktionalen Abstraktionen einer algorithmischen Sprache als Bedeutungen zuordnen kann – ist durch topologische Begriffe charakterisierbar. Ein Bezug auf algorithmische Elemente ist hierzu nicht notwendig. Während alle früheren Charakterisierungen von Berechenbarkeit *intensionale* Züge aufweisen, indem sie syntaktische Schemata zur Beschreibung von Berechnungen einführen, ist die erst in den 70er Jahren durch Scott und Strachay gefundene topologische Charakterisierung *extensional*. Sie kommt ohne linguistische Mittel zur Spezifikation von Rechenoperationen aus.⁶⁹

Konkrete Algorithmen im Sinne von Vorschriften, die präzise eine Abfolge elementarer Schritte festlegen, sind – sofern sich in Programmiersprachen überhaupt noch Elemente dieser Art finden – ungeeignete Mittel, um Verhalten zu spezifizieren. Jene Schrittfolge ist für die meisten Zwecke der Softwarekonstruktion irrelevant, weshalb man von ihr abstrahiert. Der Softwareingenieur setzt nicht elementare Operationen unmittelbar zu Systemen zusammen, sondern komponiert diese aus Abstraktionen nach Maßgabe ihrer Spezifikationen, die selbstverständlich auch konkrete technische Eigenschaften wie das Zeit- und Speicherverhalten umfassen können.

Daß man Softwaresysteme nicht unmittelbar aus elementaren algorithmischen Operationen aufbauen kann, gehört zu den fundamentalen Erkenntnissen der Informatik, die ihre Existenz als Wissenschaft der Einsicht verdankt, daß die Produktion von Software eine intellektuell herausfordernde, die Entwicklung neuer Methoden auf wissenschaftlicher Basis requirierende Tätigkeit ist. Wenn das Algorithmische das Entscheidende an der Software wäre, würde sich z. B. die Entwicklung von Programmiersprachen als reichlich nutzloser Zeitvertreib darstellen; denn als Notationen für sequentielle Algorithmen unterscheiden sich die verschiedenen Sprachen nicht nur kaum sondern sie sind auch – bisher konnte kein Sprachdesigner die These von Church widerlegen und wahrscheinlich be-

69. Die Theorie von Scott und Strachay besitzt in noch höherem Maße als die Kalküle von Schönfinkel, Church, u. a. den Nachteil, längst nicht so sinnfällig zu sein wie die Maschinenmetapher Turings. Eine knappe Skizze ohne die schwierigen Details gibt die Turing Award Lecture von Scott. Vgl. Dana S. Scott: »Logic and Programming Languages«. 1976 Turing Award Lecture. *ACM Turing Award Lectures: The First Twenty Years*. New York, NY: ACM Press, 1987, S. 47–62.

absichtigt das auch niemand – algorithmisch äquivalent. Nicht die Suche nach größerer algorithmischer Leistung motiviert die Entwicklung von Programmiersprachen, sondern die Suche nach mächtigeren Abstraktionen, die es dem menschlichen Intellekt ermöglichen, komplexe Aufgaben durch die Bildung und Auswertung von Modellen zu lösen, die aus implementationsunabhängig spezifizierten Komponenten aufgebaut sind.

Die von Heintz so groß herausgestellte Turing-Maschine ist in dieser Hinsicht unter den vielen theoretischen Modellen der Berechenbarkeit das unfruchtbarste: kein vernünftiger Informatiker wird sich daran orientieren. Als ergiebig stellten sich dagegen die von Heintz ignorierten Entwürfe von Schönfinkel, Curry, Church, Gödel und Herbrand heraus. Sie beeinflussen noch heute die Entwicklung von Programmiersprachen.

Niemand konzipiert ein komplexes Softwaresystem auf der Ebene elementarer algorithmischer Operationen. Allen wesentlichen Entwicklungsrichtungen in der Softwaretechnik ist gemeinsam, daß sie nach Mitteln suchen, um von dieser Ebene zu abstrahieren. Auf verschiedene Weise tun dies z. B. logische, funktionale und objektorientierte Sprachen.

›Abstraction provides the two key benefits of locality and modifiability. Both are based on the distinction between an abstraction and its implementation. Locality means that each implementation can be understood in isolation. An abstraction can be used without our having to understand how it is implemented, and it can be implemented without our having to understand how it is used. Modifiability means that one implementation can be substituted for another without disturbing the using programs.

To obtain these benefits, we must have a description of the abstraction that is distinct from any implementation. [...] Users can assume the behavior described by the specification, and implementers must provide this behavior. Thus the specification serves as a contract between users and implementers.⁷⁰

Der Zusammenhang eines Softwaresystems ist als Sequenz von algorithmischen Elementaroperationen – quasi analog zum Ineinandergreifen eines mechanischen Räderwerks – nicht verstehbar. Liskov und Guttag, von denen schrittmachende Beiträge zur Softwaretechnik stammen, bemühen ein Bild aus dem Rechtsverkehr zwischen Personen, um den Aufbau von Software aus abstrakt spezifizierten Komponenten zu illustrieren. Doch tatsächlich ist das mehr als nur ein Bild: Technik ist weder ein Arsenal von Artefakten noch eine Ansammlung von Algorithmen sondern Teil men- manu-
-llschlichen Handelns. Wer eine technische Lösung für eine formulierte Aufgabe bereitstellt bzw. akzeptiert, geht einen Vertrag ein, der den Parteien Rechte und Pflichten auferlegt. Der Nutzer muß die Spezifikation und die Gebrauchsanweisung beachten und darf das spezifizierte Verhalten erwarten, der Lieferant muß dieses – meist gegen Bezahlung – produzieren, wobei es ihm innerhalb der explizit formulierten Einschränkungen freigestellt ist, wie er das tut. Gerade diese Freiheit bedeutet auch, daß er *nicht*

70. Barbara Liskov, John Guttag: *Abstraction and Specification in Program Development*. Cambridge, MA: MIT Press, 1986, S. 54.

an bestimmte Algorithmen – sofern solche für die zur Diskussion stehende technische Aufgabe überhaupt von Belang sind – gebunden ist.

Nicht nur, daß das Aufkommen der formalen Kalküle die Mathematik keineswegs auf ein rein formales Hantieren mit bedeutungslosen Marken reduzierte, nicht nur, daß diese Kalküle keine Umsetzung in sequentielle Algorithmen erzwingen: gerade dort, wo die formalen Systeme technische Bedeutung gewinnen, wird die Abstraktion vom Algorithmischen praktisch unumgänglich und das hat mit dem sozialen Charakter von Technik zu tun: Wenn es darum geht, nicht bewundernswürdige Monströsitäten aufzutürmen, sondern verantwortlich Funktionen zu erfüllen, ist die Abstraktion von den mechanischen Details unerlässlich. Im Aufbau, im Zusammenspiel der Komponenten eines Artefakts, eines Prozesses, spiegeln sich – oft über mehrere Ebenen der Komposition hinweg – nicht allein naturgesetzliche Zusammenhänge sondern auch die komplementären Beziehungen von Hersteller und Verwender, von Auftragnehmer und Auftraggeber.

Es ist genau diese Konzentration auf Schnittstellen, die komplexe technische Artefakte, Prozesse und erst recht industrielle Organisationen beherrschbar macht. Nicht der Algorithmus ist das Signum rationeller Organisation und Produktion sondern die Abstraktion vom Detail – natürlich auch vom algorithmischen, sofern Algorithmen dabei überhaupt eine Rolle spielen –; wobei der Inhalt der Abstraktion nicht naturgesetzlich gegeben sondern intentional bestimmt ist.

Der mechanische, sequentielle Algorithmenbegriff von Heintz wird weder der Mathematik, noch der Informatik oder gar der sonstigen Technik gerecht. Es ist nicht so, daß sich in der Vielfalt der konkreten Maschinen die Universalität des algorithmischen Begriffs der Technik zeigte, sondern umgekehrt: der Algorithmenbegriff von Heintz ist bewußtloser Reflex des am Modell des Räderwerks gebildeten Maschinenbegriffs, seine Rigidität Residuum einer entsubstantialisierten Mechanik. Diesen Begriff projiziert sie auf die Mathematik und Informatik, um in ihnen dann – wie durch ein Wunder – das Geheimnis der Maschinenwelt zu entdecken.

Tragende Komponente des Glaubens an die Mechanisierbarkeit aller geistigen Prozesse ist die Annahme, daß man intellektuelle Operationen bis auf Elemente zerlegen könne, die einfach genug wären, damit eine Maschine sie ausführen könnte.⁷¹ Doch eine intellektuelle Operation – und sei sie noch so einfach – ist etwas anderes als eine physische. Maschinen führen physische, keine intellektuellen Operationen aus. Signifikanz erhalten die physischen Operationen und ihre Operanden erst durch die Interpretation des Intellekts. Hier liegt genau derselbe Irrtum vor wie bei der Verwechslung von formalen Objekten mit Marken auf dem Papier. Bettina Heintz verwechselt Zeichen und Zeichenträger, Denken und Kritzeln. Es

71. Heintz, a. a. O., S. 95 u. S. 284.

geht hier nicht um den Gegensatz von Komplexität und Einfachheit sondern um die Unterscheidung zwischen der intellektuellen und der physischen Ebene: Ein Gedanke ist oft viel einfacher – nicht unbedingt jedoch leichter – als das Funktionieren einer Maschine. Doch ein Gedanke ist ein Gedanke und ein Haufen Transistoren ist ein Haufen Transistoren – gleichgültig wie einfach oder kompliziert sie sein mögen.

Ein populärer Einwand gegen dieses Argument besteht in dem Hinweis darauf, daß auch das menschliche Gehirn eine Anhäufung organischer Masse sei. Das ist richtig, doch die Selbsterfahrung des Denkens weist dieses nicht als Komponente des physikalischen Zustands einer organischen Masse aus. Mit dem Denken gehen physische Vorgänge einher, ohne daß es auf solche reduzierbar wäre.⁷²

to d. nur

Technik, Arbeit und Tanz

Die Idee, das Entscheidende an der Technik liege im Algorithmischen kann vor einer genaueren Betrachtung wichtiger Techniken nicht bestehen. Welche Algorithmen – sofern Sie solche überhaupt ausführen – führen Brücken, Reaktoren, Leitungen (für Wasser, Elektrizität, Licht, etc.) aus? Wir wissen es nicht und vor allem muß man das auch nicht wissen, um funktionierende Brücken, Reaktoren, Leitungen zu bauen.

Es gibt einen Ort, wo die beschränkte technische Relevanz von Algorithmen klar zutage tritt: Nämlich dort, wo Computerprogramme eine unmittelbare technische Bedeutung haben: in der Prozeßsteuerung, bzw. der Steuerung von Ein- und Ausgabe. Während die Semantik formaler Notationen zur Spezifikation von Berechnungen empiriefrei formulierbar ist, hängen die Effekte, die Programme ausüben, von ihrer Abbildung auf die physischen Umgebung, in der sie laufen, und deren Eigenschaften ab. Natürlich verhält ein Programm sich nur dann entsprechend der Semantik, wenn diese Abbildung eine korrekte Implementation des zugrunde liegenden formalen Systems darstellt. Doch hat dies, da es eine globale Anwendungsbedingung der Implementation darstellt, keinen Einfluß auf die Gestalt von Algorithmen.

— u

Der konkrete Effekt eines Programms ist nicht durch seine Eigenschaften als formales Objekt – als solches denotiert es nur mathematische Objekte in der Semantik, keine materiellen Effekte – bestimmt, sondern durch seine Implementation. Für die äußerlich wahrnehmbaren Effekte von Programmen sind sprachliche Elemente verantwortlich, die in der formalen, empiriefreien Semantik nicht spezifizierbar sind: die Ein- und

72. Die kategoriale Differenz physischer und geistiger Bestimmungen, die gleichwohl als Attribute einer Substanz zu sehen sind, deren genauer Zusammenhang uns weitgehend unbekannt ist und wohl auch unbekannt bleiben wird, gehört immer noch zu den denkwürdigen Errungenschaften der Moderne. Vgl. Baruch Spinoza: *Ethica. Ordine Geometrico demonstrata*. Pars tertia, Propositio ii.

Ausgabeeweisungen. Diese gehören nicht zu einer formalen algorithmischen Sprache. Da man einer algorithmisch vollständigen Sprache nichts hinzufügen kann, was ihre algorithmische Potenz erhöhen würde, und solche Anweisungen auch nicht die Formulierung von Algorithmen erleichtern, ist kein anderer Schluß möglich. Viele Programmiersprachen umfassen deshalb auch keine derartigen Anweisungen sondern ihre Implementationen stellen Ein- und Ausgabe als Module bereit, die als symbolische Verpackungen für Hardwarefunktionen zu verstehen sind.

Der Effekt der Ausführung einer bestimmten Ein- oder Ausgabeoperation wie z.B. von `>output (std_out, "hi folks")<` hängt von der besonderen Konfiguration von Systemsoftware und Hardware ab, die die Umgebung des Programmlaufs bildet. Die Beschreibungen von Programmiersprachen nehmen meistens Bezug auf eine Standardumgebung, die natürlich historischen Veränderungen unterliegt. Vor drei Jahrzehnten hatte eine Ausgabe auf `>std_out<` noch einen ganz anderen physischen Effekt als heute! Manifest wird die Abhängigkeit von der Hardwareumgebung bei graphischer Ausgabe: Ob man ein Graphiksubsystem mit 1 MByte Video-RAM und 15"-Schirm für 256 Farben oder eines mit 4 MByte Video-RAM und 20"-Schirm, der 16 Millionen Farben differenzieren kann, benutzt, macht einen Unterschied, der nicht durch verschiedene Algorithmen erklärbar ist. Es ist also nicht so, wie der Heintzsche Technikbegriff nahelegt, daß die wahrnehmbare Funktion der Maschine Computer durch einen Algorithmus determiniert wäre: Den Effekt eines Programms bestimmen ebenso dessen unalgorithmische Elemente – die Ein- und Ausgabeeweisungen – und der physische Aufbau der Maschine.

Ein Programm kann dazu dienen, eine spezifizierte technische Aufgabe zu lösen, etwa die Regelung einer Größe, die durch Sensoren meßbar und durch Effektoren beeinflußbar ist, wie die Temperatur in einem Gefäß. Doch bilden die algorithmischen Komponenten des Programms, die zudem viele funktional äquivalente Alternativen zulassen, dabei nur einen Faktor unter vielen und vor allem hängt die Wirkung bestimmter Operationen des Programms davon ab, daß die Hardware eine ganz bestimmte Konstruktion aufweist: Es müssen gewisse Adressen auf Steuer- bzw. Datenregister von speziellen Subsystemen, nämlich den Schnittstellen zu den Sensoren und Effektoren abgebildet sein, diese wiederum gewisse Eigenschaften besitzen, Interruptvektoren auf Adressen zeigen, auf denen bestimmte Programmkomponenten liegen, etc. Die Gestalt des Steuerprogramms richtet sich nach den Eigenschaften der Hardware und nicht umgekehrt. Noch nicht einmal eine Prozeßsteuerung ist rein algorithmisch zu verstehen; noch viel weniger eine beliebige Maschine.

Die Pragmatik einer Steuerung liegt ebenso wie die physische Funktion einer Maschine generell außerhalb des Bereichs der Algorithmen und ihrer Semantik. Algorithmen sind Verfahren zur Transformation von for-

malen Objekten, deren Semantik empiriefrei ist. Sofern die Ausführung eines Algorithmus außersymbolische, materielle Effekte hat, hängen diese mit der materiellen Repräsentation symbolischer Objekte in der Hardware zusammen. Diese Effekte gehören nicht zur semantischen Spezifikation von Algorithmen. Heutige Universalrechner, Betriebs- und Programmersysteme unterstützen bestimmte standardisierte externe Effekte in Form von Ein- und Ausgabeeinweisungen. Spezielle Effekte, wie sie der Betrieb von Steuerungen erfordert, bedürfen auch spezieller Hardware und vor allem ist die Beschreibung der Effekte eines Steuerungsprogramms ohne eine Kenntnis dieser Hardware bzw. ihrer Spezifikation nicht möglich. Die Wahrheit einer Zusicherung über den Zustand dieser Hardware wie z.B. »Überdruckventil offen« hängt davon ab, daß eine Vielzahl von Ceteris-paribus-Klauseln wie z.B. »die Stromzufuhr zum Effektor ist nicht unterbrochen« etc. erfüllt ist. Die Algorithmen sind keinesfalls das Entscheidende an einer Steuerung sondern durch funktional äquivalente austauschbare, untergeordnete Komponenten. Algorithmische Elemente in Steuerungen sind zudem oft durch Hardwarefunktionen ersetzbar, wie z.B. eine Schleife durch einen Timer-Interrupt.

Die Austauschbarkeit von algorithmischen Elementen durch Hardwarefunktionen resultiert nicht aus einer Kongruenz von Algorithmus und Maschinerie sondern aus der Tatsache, daß Steuerungsprogramme sich den Umstand zunutze machen, daß sequentielle Maschinen Algorithmen in der Zeit fortschreitend ausführen und die Einbettung einer Anweisung in eine Schleife deren Wiederholung in der Zeit und nicht etwa deren parallele Vervielfältigung bedeutet. Sie benutzen das algorithmische Konstrukt der Schleife nicht als solches, sondern um eine Eigenschaft seiner Implementation auszunutzen. Ob das den gewünschten Effekt bringt, hängt von physikalischen Parametern ab wie der Zeit, in der die Hardware die Schleife abarbeitet. Während die Korrektheit eines Algorithmus niemals von der Geschwindigkeit abhängt, mit der er ausgeführt wird, kann dies bei einem Steuerungsprogramm durchaus der Fall sein.

Programme

Zwar ist es üblich, bei der Komplexität von Berechnungen von Zeit zu reden, doch geht es hierbei nicht um die physikalische Zeit. Daß man eine Berechnung auch als Baum sehen kann, dessen Knoten auf verschiedene Weise in zeitlicher Folge besuchbar sind, wurde bereits erwähnt. Aussagen über den Zeitaufwand der Berechnung gehen dann über in Aussagen über die Zahl der Knoten des Baumes und solche über die Zeitkomplexität eines Algorithmus in solche über die Abhängigkeit der Knotenzahl in der ihm zugeordneten Klasse von Bäumen von seinen Parametern. Bei Algorithmen ist diese Zahl bzw. die Zeitkomplexität eine Funktion ihrer Parameter, bei Steuerprogrammen jedoch nicht, da sie dort von externen Ereignissen abhängt. Die meisten Steuerprogramme sind als Endlosschleifen konzipiert - also algorithmisch völlig unbestimmt.

Daß der Algorithmus nicht das Entscheidende an einer Steuerung ist, ist lediglich eine Konsequenz aus der bereits dargelegten Tatsache, daß der sogenannte Universalrechner *keine* universelle Maschine ist, sondern eine spezielle Maschine zur Realisierung bestimmter Klassen von formalen Systemen. Vor allem ist die Realisierung eines formalen algorithmischen Systems selbst kein Algorithmus. Soll ein Universalrechner eine Rolle innerhalb einer Maschine anderer Art übernehmen, muß man seine Hardware erweitern. Es ist deshalb auch kein Wunder, daß die Funktion dieser Hardwareerweiterungen innerhalb des Rahmens der formalen Systeme, die der Universalrechner implementiert, nicht beschreibbar ist.

Das Universum der Maschinen ist Bestandteil der physischen und sozialen Welt. Maschinen sind *keine* Verkörperungen von Algorithmen, da diese ihrem Wesen nach keine Bestimmungen enthalten, die solche von physischen Gebilden und der Funktionen, die sie im sozialen Gebrauch anzunehmen vermöchten, sein könnten. Allerdings gibt es gewisse Analogien zwischen den Methoden, die zur Konstruktion von Softwaresystemen taugen und denen, die beim Aufbau von technischen Systemen anzuwenden sich empfehlen: Beiden Arten ist die Abstraktion den Details des mechanischen Ablaufs gemeinsam.

Auch die Organisation einer Fabrik läßt sich nicht top-down bis hin zum Ineinandergreifen der letzten Handgriffe planen. Das Resultat wäre viel zu starr und kaum noch dazu in der Lage, mit unvorhergesehenen Zwischenfällen oder neuen Anforderungen fertig zu werden. Nach teuren Fehlschlägen mit CIM und der vollautomatischen Fabrik erhält in der Industrie die keinesfalls neue Erkenntnis wieder mehr Gewicht, daß es nicht immer sinnvoll ist, jeden Handgriff zentral zu programmieren. Es ist oft viel effektiver, bestimmte Entscheidungen lokal zu treffen. Das Konzept einer nach starren Regeln mechanisierten Massenproduktion ist zudem nur attraktiv, solange es einen unbegrenzten Markt für gleichförmige Produkte gibt. Das ist jedoch nur selten der Fall.

Komplementär zu dem überspannten Bild von der Fabrik als Algorithmus ist das bildungsspießbürgerliche Ressentiment gegen die ›geistlose Fabrikarbeit‹.⁷³ Beide zusammen legen den Verdacht nahe, daß die Technikerfahrung der Autorin nicht übermäßig profund sein kann. Daß Fabrikarbeit eintönig und belastend sein kann, ist nicht zu bezweifeln, doch daß sie generell geistlos wäre oder gar sein müßte schon.

Bettina Heintz kommt nur deshalb auf die Idee, daß die nach starren Regeln bis zum letzten Handgriff programmierte Produktion ein algorithmischer Prozeß sei, weil ihr Begriff von einem formalen System der Berechnung einerseits viel zu eng ist und andererseits sachfremde Elemente enthält: Sie hält die bildhafte Ähnlichkeit der Turingmaschine mit einer vage mechanische Züge tragenden Apparatur für deren theoretische

73. Heintz, ›Das Fließband im Kopf‹, a. a. O., S. 50.

Substanz. Doch genau jene Ähnlichkeit ist daran kontingent und völlig irrelevant – und zwar nicht nur, wenn es darum geht, den Gehalt und Charakter der modernen Mathematik aufzufassen, sondern auch, wenn es um einen adäquaten Begriff von Technik geht.

Auf dem Umweg über die Fälschung ›mathematischer Formalismus‹ und eine reichlich mißverstandene Informatik erhebt Bettina Heintz eine kontingente Gestalt von Technik zu deren Essenz. Merkwürdigerweise nimmt sie jedoch, kaum daß sie behauptet hat, daß das Entscheidende am Technischen das Algorithmische sei,⁷⁴ den theoretischen Anspruch, etwas zum Verständnis von Technik geleistet zu haben, wieder zurück:

›Ähnlich wie der methodologische Funktionalismus beinhaltet der vorgeschlagene Technikbegriff nur ein methodisches Prinzip. Er macht keine Aussagen darüber, was technische Artefakte »sind«, sondern behauptet nur, daß nicht die Materialität eines Artefakts die angemessene Beschreibungsebene ist, sondern die funktionale Organisation. Alles, was sich in Termini eines algorithmischen Prozesses *beschreiben* läßt, fällt unter diesen Technikbegriff. Für welche Artefakte bzw. technische Systeme dies im einzelnen gilt, läßt sich nicht generell beantworten – obschon die neuen Informationstechniken dafür vermutlich die besseren Kandidaten sind.«⁷⁵

Das ist Chuzpe: Zuerst großspurig den neuen, revolutionären Technikbegriff ankündigen, ihn dann langatmig ausbreiten und schließlich erklären, daß eigentlich noch unklar sei, was unter ihn falle. Hier liegt eine neue Variante jener postmodernen Immunisierungsstrategie vor, der zufolge die Welt ohnehin in selbstreferentielle Diskurse zerfalle, eine kohärente Bezugnahme auf Realität sich also erübrige. Möglicherweise spielt es auch keine Rolle, daß die Extension der schicken Begriffskonstrukte leer ist.

Genau letzteres ist beim algorithmischen Technikbegriff der Fall: Anders als Bettina Heintz vorgibt, kann man nämlich genau sagen, was unter ihn fällt: *nichts*. Ein Raisonment über Beschreibungsebenen, das dunkel auf die Unangemessenheit der ›Materialität‹ – was immer das sei und wer auch immer sich dort herumtreibe – als solche verweist und nicht wahrnimmt, daß die funktionale Organisation eines Artefakts etwas anderes ist als ein Algorithmus, ist überflüssig.

Beeindruckend auch die Sorglosigkeit, mit der Bettina Heintz weitreichende Behauptungen aufstellt. Darunter fällt auch die Gedankenlosigkeit, daß Formalisierung und Mechanisierung ›bedeutungsäquivalente‹ Prozesse bzw. Begriffe seien.⁷⁶ Zunächst ist unklar, was sie mit ›bedeutungsäquivalent‹ meint: ›Bedeutungsgleich‹ kann sie nicht meinen, denn daß die Begriffe verschiedene Bedeutungen haben, ist offenkundig. Wahrscheinlich ist ›extensional äquivalent‹ gemeint. Bekanntlich sind z. B. die beiden Begriffe ›vernunftbegabtes Sinnenwesen‹ und ›ungefiederter Zwei-

74. Heintz, *Die Herrschaft der Regel*, a. a. O., S. 258.

75. A. a. O., S. 259.

76. A. a. O., S. 11 und 94.

Leibniz Arbeit

Körperliche Arbeit

↓
Algorithmus

↓
Abstraktion

Die Mechanisierung
körperlicher Arbeit

38

↓
Mechanismus

RAINER FISCHBACH

requisit - anders als die
Mühle - keinen Algorithmus

beiner - zumindest im uns bekannten Teil des Universums - koextensional; was nicht heißt, daß ein vernunftbegabtes Sinnenwesen zu sein das gleiche bedeute wie ein ungefiederter Zweibeiner zu sein.

Doch nicht einmal die Koextensionalität ist gegeben: Seit Jahrhunderten mechanisieren Menschen Tätigkeiten wie das Mahlen von Korn, das Schmieden von Stahl oder das Zersägen von Baumstämmen in Bretter, ohne die geringste Vorstellung von deren Formalisierung zu besitzen. Andererseits requiriert die Mechanisierung formaler Systeme die Existenz geeigneter Algorithmen. Es gibt in der Mathematik genügend präzise formulierbare Probleme, deren Formalisierung zwar wunderschön aussieht, uns ihrer Lösung jedoch keinen Schritt näher bringt, weil es keinen Algorithmus gibt, der ihre Mechanisierung erlaubt. Wie bereits angedeutet, kann man z. B. die Semantik von algorithmischen Sprachen formalisieren; was jedoch nicht impliziert, daß wir sie auch mechanisieren könnten. Es gibt u. a. keine mechanische Lösung des Problems der extensionalen Äquivalenz von Abstraktionen und des Halteproblems, weil diese algorithmisch unentscheidbar sind.

Von nicht minderer Leichtfertigkeit zeugen die kulturhistorischen Thesen von Frau Heintz. Weshalb in den Tanzrevuen der 20er Jahre das Rationalisierungsprinzip des beginnenden 20. Jahrhunderts zum Ausdruck kommen soll,⁷⁷ ist schwer nachvollziehbar. Daß »Bewegungen gemessen, analysiert, in ihre Bestandteile zerlegt und nach rationalen oder ästhetischen Prinzipien neu zusammengefügt«⁷⁸ werden, ist nicht erst dort der Fall: das taten schon die Begründer des akademischen Tanzes im 17. Jahrhundert⁷⁹ und vor ihnen Moritz von Nassau, der Erfinder des modernen militärischen Exerzierreglements.⁸⁰

Welche Zusammenhänge existierten zwischen der Herrschaft über die Körper in den damals aufkommenden Manufakturen und der Beherrschung des Körpers im Tanz arbeitete vor mehr als zwei Jahrzehnten schon Rudolf zur Lippe heraus.⁸¹ Daß zu deren Klärung mehr erforderlich war als ein algorithmischer Begriff von Technik, ist seinem Werk zu entnehmen. Daß gerade die Beherrschung des Körpers immer auch eine Auseinandersetzung mit dessen immanenter Konstitution und Gesetzmäßigkeit bedingt - die Positionen sind nicht beliebig kombinierbar wie die Operationen einer Turingmaschine -, entlarvt den algorithmischen Technikbegriff vollends als erfahrungslose Fiktion.

77. Heintz, »Das Fließband im Kopf«, a. a. O., S. 54.

78. A. a. O.

79. Vgl. Gerhard Zacharias: *Ballett - Gestalt und Wesen: Die Symbolsprache im europäischen Schautanz der Neuzeit*. 2. Aufl., Wilhelmshaven: Noetzel, 1993, S. 56 ff.

80. Vgl. William H. McNeill: *Krieg und Macht: Militär, Wirtschaft und Gesellschaft vom Altertum bis heute*. München: Beck, 1984, S. 118 ff.

81. Vgl. Rudolf zur Lippe: *Naturbeherrschung am Menschen*. Frankfurt am Main: Suhrkamp, 1974.

Selbstverständlich sind auch die Operationen realer Maschinen nicht beliebig kombinierbar: Sie unterliegen vielfältigen Einschränkungen. Wirkliche Technik schließt die Auseinandersetzung mit Materie, mit der Natur, mit dem menschlichen Körper, dem menschlichen Geist ein. Dies bedeutet auch, auf kontingente Bedingungen und Ereignisse einzugehen. Nach aller bisheriger Erfahrung erfordert dies immer noch Wachheit, Sensibilität und Phantasie.

Bettina Heintz phantasiert Technik dagegen als Ritual: Auf nichts anderes läuft ihr algorithmischer Technikbegriff hinaus, der immer wieder die blinde Ausführung rigider Sequenzen von elementaren Operationen betont. Diese Emphase ist weder besonders rational noch besonders modern: Sie ist geradezu ein Kennzeichen traditioneller Gesellschaften. Sybille Krämer sieht hierin einen möglichen Grund für die hohe Bedeutung von Algorithmen in Gesellschaften wie der des antiken Ägypten:

›Das mathematische Wissen ist ein Regelwissen, das seine Sicherheit dadurch gewinnt, daß eine mathematische Handlung als starres Einhalten und praktischer Vollzug eines überkommenen Schemas gilt. Könnte es hier einen Zusammenhang geben zum traditionellen, ritualisierten Grundzug dieser Gesellschaften, in denen sozialer Sinn von Handlungen gestiftet wird durch die Einhaltung vorgegebener Schemata?‹⁸²

Diese nicht unplausible Vermutung widerspricht allerdings der These von Heintz, daß sich im Siegeszug des Algorithmus die Rationalisierung und Modernisierung der westlichen Gesellschaften vollziehe. Was an dieser Rationalisierung Fiktion einer überspannten Theoriebildung und was Realität ist, bedarf ohnehin noch genauerer Untersuchung. Z. B. Max Webers Vergleich eines modernen Richters mit einem Automaten, den Heintz zitiert, ist nicht besonders überzeugend: Wenn die Richter der Weimarer Republik so rational und automatisch gearbeitet hätten, hätten Adolf Hitler sowie viele seiner Kumpanen und Wegbereiter das Jahr 1933 nicht in der Reichskanzlei bzw. in Freiheit sondern im Zuchthaus erlebt. Leider hatte die Weimarer Justiz dieses Maß an Automatisierung noch nicht erreicht.

Wie schwierig es ist, Realität und Propaganda auch in der industriellen Rationalisierung in jener Zeit auseinander zu halten, bestätigen die Untersuchungen von Joachim Radkau.⁸³ Woher Hilbert, Turing u. a. die sozialen Erfahrungen, die sie zur konsequenten Übertragung des Mechanisierungs- und Rationalisierungsparadigmas auf ihre wissenschaftliche Arbeit motiviert hätten, genommen haben sollten, bleibt unklar. Fließband und Taylorismus bestimmten zwar eine Zeitlang die öffentliche Diskussion, doch entsprach diese Diskussion nicht dem Stand der Umsetzung. Möglicherweise war das damals mit diesen Begriffen nicht viel anders als heute mit ›Informationsgesellschaft‹, ›Multimedia‹ und ›Cyberspace‹.

82. Sybille Krämer, a. a. O., S. 178 f.

83. Joachim Radkau, a. a. O., S. 269 ff.

Die Heintzsche Theoriebildung zu Formalisierung, Metamathematik und Computertechnik geht von der Diskussion *über* diese Dinge bzw. von deren Niederschlag in der Sekundärliteratur aus und nicht von einem Verständnis ihres wissenschaftlichen Gehalts. Die Äußerungen von Wissenschaftlern über ihre Positionen oder die ihrer Diskussionsgegner sind oft taktischer Natur. Das trifft nicht zuletzt auch auf die Auseinandersetzung um die Grundagentheorie zu. Es ist jedoch völlig vermessen, aus einzelnen Stellungnahmen und Argumentationsgängen die wissenschaftliche Substanz des Werkes von Mathematikern wie Hilbert, Weyl und Brouwer erschließen zu wollen. Wenn Hilbert in einer bestimmten Diskussion eine formalistische Position bezog oder Weyl ihm eine solche zum Vorwurf machte, heißt dies noch längst nicht, daß diese sein gesamtes Werk auszeichne oder gar, daß er damit etwas Abschließendes über die Weise, wie Mathematik zu betreiben wäre, gesagt hätte.

Große Wissenschaftler sind nicht immer die klarsichtigsten Wissenschaftstheoretiker. Davon machen Leute wie Hilbert, Brouwer, Einstein u. a. keine Ausnahme. Auch sie neigen dazu, lokale Debatten zu globalisieren. So war es üblich, formale metamathematische Techniken als Versuche zu diskutieren, die ganze Mathematik auf diese Weise zu betreiben. Diese hat sich jedoch glücklicherweise dagegen als resistent erwiesen: Die Metamathematik – also das Studium formalisierter Theorien – ist heute als eine Disziplin der intuitiven Mathematik neben anderen etabliert und der Intuitionismus als ein bestimmter, durchaus auch der Formalisierung fähiger Stil, Mathematik zu betreiben, ebenfalls. Eine konsequent formalistische mathematische Praxis gibt es eigentlich nicht. Man kann das heute also alles viel entspannter sehen.

Es gibt keinen Grund, die Aufgeregtheit der damaligen Debatten wieder aufleben zu lassen und erst recht keinen, einen verdienten Mathematiker wie Brouwer, der wie viele Intellektuelle der Weimarer Zeit reaktionären lebensphilosophischen Positionen nahestand, als Vorreiter der Anti-Moderne zu portraituren, in dessen Mathematik jene philosophischen Positionen ihren Ausdruck gefunden hätten.⁸⁴ Manche Passagen bei Heintz wirken wie Argumentationen *ad hominem* bzw. *ad feminam*, die auf ein irgendwie sich progressiv oder feministisch verstehendes Publikum zielen.⁸⁵ Die Überzeugungen eines Menschen sagen nur bedingt etwas über den Wert seiner Arbeit aus. Brouwer stellte valide mathematische Fragen und versuchte radikale Antworten darauf zu geben, denen man nicht zustimmen muß, jedoch die Diskussionswürdigkeit nicht abstreiten kann. Daß ihnen heute in der theoretischen Informatik wachsende Bedeutung zukommt, unterstreicht diese Diskussionswürdigkeit. Frau Heintz glaubt dagegen, folgendes feststellen zu müssen:

84. Heintz, *Die Herrschaft der Regel*, a. a. O., S. 197.

85. Vor allem a. a. O., S. 43.

»Die »moderne« Mathematik hatte sich als Arbeitsprogramm durchgesetzt, und der Intuitionismus ist für die Nachgeborenen bloß noch eine kurze Episode in der Geschichte der Mathematik und ihrer Krisen.«⁸⁶

Dieses hingerotzte Statement bezeugt vor allem die Ahnungslosigkeit seiner Urheberin. Doch außer den reichen Beiträgen zur inhaltlichen Mathematik besteht das große Verdienst der intuitionistischen Schule, von Brouwer, Poincaré, Weyl, Lorenzen, u. a. darin, besonders eindringlich nach den Prinzipien der wissenschaftlichen Praxis und deren Begründung gefragt zu haben. Das ist – jenseits von dogmatischen Grundlegungsprogrammen – immer noch eine moderne und höchst rationale Tugend, die nicht nur in der Mathematik sondern auch in der Technik angezeigt ist. Daß das nicht ganz im Trend der zeitgeistigen Beliebigkeit liegt, unterstreicht nur die Dringlichkeit, auf die Bedeutung moralischer Prinzipien der wissenschaftlichen Argumentation hinzuweisen.⁸⁷

86. A. a. O., S. 200.

87. Vgl. Paul Lorenzen: »Moralische Argumentationen im Grundlagenstreit der Mathematiker«. Paul Lorenzen: *Methodisches Denken*. Frankfurt am Main: Suhrkamp, 1968 (stw; 93), S. 152–162.

